Tommi Wiik

Graafisten pelielementtien tuotantolinja

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

25.4.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Wiik Tommi Graafisten pelielementtien tuotantolinja 45 sivua 25.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Toimitusjohtaja Toni Nylund Lehtori Antti Laiho

Insinöörityön tarkoituksena on selvittää, miten pelin graafisten elementtien tuotantolinja kannattaa rakentaa ja mitä siinä tulee ottaa huomioon, ja toteuttaa se käytännössä. Insinöörityö tehtiin itsenäiselle mobiilipelistudiolle.

Tuotantolinjan käsite on läsnä kaikkialla, ja niitä käytetään jatkuvasti perinteisestä teollisuudesta arkielämään. Peliteollisuudessa tuotantolinjoista on muodostunut tärkeitä projektinhallinnassa käytettäviä työkaluja, jotka ohjaavat tuotetun resurssin laatua, määrää ja yhteensopivuutta keskenään. Kun tuotantolinjan määrittämistä projektille aloitetaan, on tärkeää perustaa tehtävät päätökset projektin perustaan eli pelin suunnitelmaan. Pelin suunnitelma, julkaisualusta ja pelimoottori määrittävät tuotantolinjalle reunaehdot, joiden sisällä sen tulee toimia. Tällä tavoin vältytään monelta ongelmatilanteelta, joiden korjaaminen saattaa vaatia tuotantolinjan rakenteen korvaamista.

Yrityksen pitkäaikainen projekti on mobiililaitteille suunnattu moninpeli, joka yhdistää tukikohdan suojelemista sosiaalisten ominaisuuksien kanssa laajalle yleisölle tarkoitetussa pelissä. Pelin tuotantolinja on monivaiheinen ketju työvaiheita, ja se on kokenut huomattavan määrän muutoksia. insinööriyössä muodostettu tuotantolinja on optimoitu projektin tämänhetkisiin tarpeisiin ja se helpottaa pelielementtien tuotantoa huomattavasti. Yhtenäistetyt tuotantomääritteet auttavat pitämään pelielementtien ulkonäön ja toiminnallisuuden johdonmukaisena ja nopeuttavat tuotantoprosessia. Tuotantolinjaa kehitetään eteenpäin projektin tarpeiden mukaan sen edetessä. Muutosten tekeminen ja iterointi ovat olennainen osa pelinkehityksessä usein käytettävää ketterää kehitysmallia.

Avainsanat	pelinkehitys, 3D-mallinnus, teksturointi, tuotantolinja, 3D- ja 2D-grafiikka, pelielementit

Author Title Number of Pages Date	Tommi Wiik The pipeline of graphical game assets 45 pages 25 April 2016
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Toni Nylund, CEO Antti Laiho, Senior Lecturer

The purpose of this thesis is to determine how a pipeline for graphical game assets should be set up and what variables must be taken into account in the process and how the pipeline should be implemented. The thesis was done for an independent mobile game development studio.

The concept of a pipeline is present everywhere and it is used from conventional industry to everyday life. In the game industry pipelines have become an important part of the project management toolset that maintains the quality, quantity and compatibility of the products with each other. When the pipeline is being defined for a project, it is of major importance that the decisions are based on the foundation of the game also known as game design. The game design, target platform and game engine set the boundary conditions for the project which must be followed. This way many problematic situations can be averted which would otherwise lead to modifications of the pipeline. The company's unreleased title is a multiplayer game meant for mobile devices. It combines elements of a base defense game with social features combined in a game designed for a vast audience.

The asset pipeline of the multiplayer game is a long sequence of work stages and it has seen a large number of modifications. The practical work done in the final year project resulted in a pipeline optimized for the project at its current state and has made asset creation considerably easier for the team. Unified production guidelines improve the consistency of the functionality and aesthetic appearance of the assets. In the future, the pipeline will be modified to suit the needs of the project as it progresses. Modification of the pipeline and iteration are a terminal part of the agile development model often used for game development.

Keywords	game development, 3D-modeling, texturing, pipeline, 3D graphics, 2D graphics, game assets

Sisällys

1 Johdanto

Insinöörityön tarkoituksena on määritellä peleissä käytettävien graafisten pelielementtien tuotantolinjan merkitys kehityksessä, pohtia, miten se kannattaa rakentaa, ja myös toteuttaa se käytännössä. Aihetta käsitellään enimmäkseen mobiilipelien näkökulmasta, mutta myös muille alustoille kehitettäviin peleihin otetaan kantaa. Työ ei kata muiden kuin graafisiin elementteihin ja niiden tuotantolinjoihin liittyviä asioita.

Työ tehdään Vulpine Games Oy:lle, joka on sosiaalisiin mobiilipeleihin keskittynyt pelistudio. Sen kehityksessä oleva esikoispeli Last Planets on massiivinen moninpeli, joka sekoittaa tukikohdan puolustusta sosiaalisten pelien elementtien kanssa. Peli on kehitetty Unity 3D -pelimoottorilla, ja se julkaistaan IOS- ja Android-mobiilimarkkinoille. Työ koostuu Last Planets -pelin tuotantolinjan määrittämisestä, sen rakentamisesta ja käyttöönotosta.

Peleissä, joissa on paljon graafisia elementtejä, kuvia ja efektejä, visuaalinen yhtenäisyys on haastavaa saavuttaa. Sama pätee myös pelissä käytettyjen elementtien valmistuksen tehokkuudessa, iteroinnissa ja nopeudessa. Tarvittavan ammattitaidon lisäksi johdonmukainen ja optimoitu tuotantolinja on avaintekijä projektin onnistumisessa. Insinöörityössä käydään läpi tuotantolinjaan vaikuttavia tekijöitä, joista tärkeimpiä ovat pelin suunnitelma, tekniset rajoitteet ja optimointi.

Tuotantolinjojen rakentaminen on tapauskohtaista, ja eri lähestymistavat toimivat eri tavalla erilaisissa projekteissa ja työympäristöissä. Osa tässä insinöörityössä läpi käytävistä asioista ovat Last Planets -projektille ominaisia, ja niiden soveltaminen muihin projekteihin kannattaa tehdä harkiten.

Last Planets -pelin tuotantolinjan rakentamisen tavoitteena oli luoda toimiva kokonaisuus, joka vähentäisi pelielementtien valmistamiseen kuluvaa aikaa ja parantaisi niiden laatua ja yhtenäisyyttä visuaalisesti ja toiminnallisesti. Käytännössä tämä tarkoitti yhteisten määritelmien luomista ja elementtien tuotannossa käytettävien ohjelmien mahdollista vaihtamista yhteensopivuuden lisäämiseksi ja aikaisemman kokemuksen hyväksikäyttämiseksi. Kokonaisuudessaan tuotantolinja ja siihen liittyvät tekijät määriteltiin ja käyttöönotettiin koko työryhmän avulla. Vaikka varsinaisen tuotantolinjan rakentamisen tekevät työryhmän graafisten pelielementtien kanssa työskentelevät henkilöt, muiden työryhmän jäsenten tarpeet ja ehdotukset otetaan huomioon.

2 Tuotantolinjojen rooli pelikehityksessä

2.1 Tuotantolinja käsitteenä ja nykypäivänä

Kun tarkastellaan tuotetta, hyödykettä, esinettä tai mitä tahansa muuta olemassa olevaa asiaa, on loogista olettaa, että se on kulkenut tiettyjen työvaiheiden läpi päästäkseen nykyiseen muotoonsa. Esimerkiksi puisen tuolin valmistuksessa puuseppä työstää raakoja materiaaleja erilaisilla työkaluilla päästäkseen haluttuun lopputulokseen. Työvaiheet, joiden läpi tuote kulkee, määrittävät sen lopullisen muodon tai ominaisuudet. Mikäli työvaiheita ei ole määritelty, lopputulokset voivat olla hyvin vaihtelevia. Jotta lopputuloksista saataisiin yhdenmukaisia, täytyy valmistettavalle asialle tai esineelle määritellä tuotantolinja.

Tuotantolinja voidaan määritellä ketjuna työvaiheita, joiden sisältö on määritelty halutulla tarkkuudella ja joiden olemassaolo on perusteltua halutun lopputuloksen kannalta. Jokainen työvaihe koostuu prosesseista, joita voidaan ajatella työvaiheen sisäisinä työvaiheina. Niiden toteutuksen tarkkuus, laajuus, kesto ja muut työvaihekohtaiset muuttujat riippuvat halutusta lopputuloksesta ja niiden vaikutuksesta seuraavaan työvaiheeseen. Ammatillisessa käytössä tuotantolinjan rakenne määritellään mahdollisimman tarkasti johdonmukaisten tulosten aikaansaamiseksi. Etenkin automatisoiduilla tuotantolinjoilla, esimerkiksi autotehtaissa, työvaiheet ja niiden prosessit on määritelty erittäin tarkasti tuotantolinjasuunnitelmassa. [1.]

Mitä kauemmas perinteisestä teollisuuden alasta (autoteollisuus, paperiteollisuus jne.) liikutaan, sitä vaikeampaa tuotantolinjan tarkka määrittely on. Hyvänä esimerkkinä tästä on graafinen suunnittelu, jossa työvaiheiden prosessien tarkka määrittely voi olla lähes mahdotonta [2]. Usein tuotantolinjalle määritetyt ohjeet rajaavat, miltä jokin näyttää tai minkälaisia visuaalisia elementtejä kuvassa tulisi olla. Tarkkoja arvoja voidaan käyttää eri värien ja kuvien resoluutioiden määrittämiseen, mutta suuri osa määritelmistä riippuu tekijän subjektiivisesta näkemyksestä.

Tuotantolinjoja käytetään kaikkialla, ja voidaan ajatella, että niitä on käytetty ihmisen historian alusta asti. Periaatteessa minkä tahansa sarjan valmistus, joka vaatii useita työvaiheita, voidaan lukea tuotantolinjaksi. Nykypäivänä tuotantolinja sanana saattaa muistuttaa ihmisiä lähinnä autotehtaiden tai muiden tuotantolaitosten tehostetuista tuo-

tantomenetelmistä, mutta todellisuudessa tuotantolinjat ovat läsnä kaikkialla. Vaikka voidaan ajatella, että tuotantolinjoja on käytetty jo tuhansia vuosia, varsinaiset teolliset tuotantolinjat alkoivat yleistyä 1900-luvun alkupuolella teollisen vallankumouksen myötä [3].

2.2 Tuotantolinjat ja niiden tyypit peliteollisuudessa

Pelit ovat digitaalisina mediatuotteina haastavia kehittää, sillä ne koostuvat usein useista eri elementeistä, joiden täytyy toimia saumattomasti hyvän kokonaisuuden saamiseksi. Eri elementtien laajuus ja vaativuus riippuvat hyvin paljon pelin tyypistä, tuotantoarvoista ja tavoitteesta. Esimerkiksi suuren mittakaavan ja budjetin open world -peleihin tarvitaan usein erittäin paljon graafisia elementtejä, musiikkia, ääniä, toiminnallisuutta ja pelimekaniikkaa, joiden valmistamiseen voi mennä vuosia, kun taas pienen mobiilipelin elementtien valmistamiseen voi kulua vain muutama päivä. Pelien mittakaavasta tai vaativuudesta huolimatta jokaisen elementin valmistamiseen tarvitaan erityistä osaamista ja niiden toteuttamiseen eri toimintatapoja. [4.]

Yksi laadukkaan pelituotteen tärkeimmistä ominaisuuksista on eri elementtien yhtenäisyys. Tämä voidaan saavuttaa hyvän suunnittelun ja yhtenäisen elementtien kehityskulun eli tuotantolinjojen avulla. Ennalta määrätyt ohjeet kunkin elementtityypin valmistamiseen takaavat oikein käytettynä yhtenäisiä lopputuloksia. Hyvin tehty tuotantolinja mahdollistaa elementtien iteroinnin ja viimeistelyn, ja sen kanssa on myös mukavampaa ja helpompaa työskennellä [5].

Kaikkia elementtityyppejä ei voi kuitenkaan kehittää saman tuotantolinjan avulla, vaan jokaiselle eri tyypille tulee muodostaa oma tarkkaan määritelty linja prosesseja, joiden läpi elementti kulkee. Tuotantolinjat voidaan karkeasti jakaa neljään eri kategoriaan:

- graafisten elementtien tuotantolinjat
- ääni- ja musiikkielementtien tuotantolinjat
- ohjelmointiin liittyvät tuotantolinjat
- pelimekaniikkaan ja designiin liittyvät tuotantolinjat.

Graafisten elementtien tuotantolinjojen rakenteet voivat vaihdella huomattavasti riippuen pelin tyypistä, alustasta ja tyylistä. Esimerkiksi eroavaisuuksien määrä 3D- ja 2D-pelien tuotantolinjoissa on usein huomattava. Tämä johtuu siitä, että niiden valmistamiseen tarvittavat työaskeleet eroavat toisistaan merkittävästi.

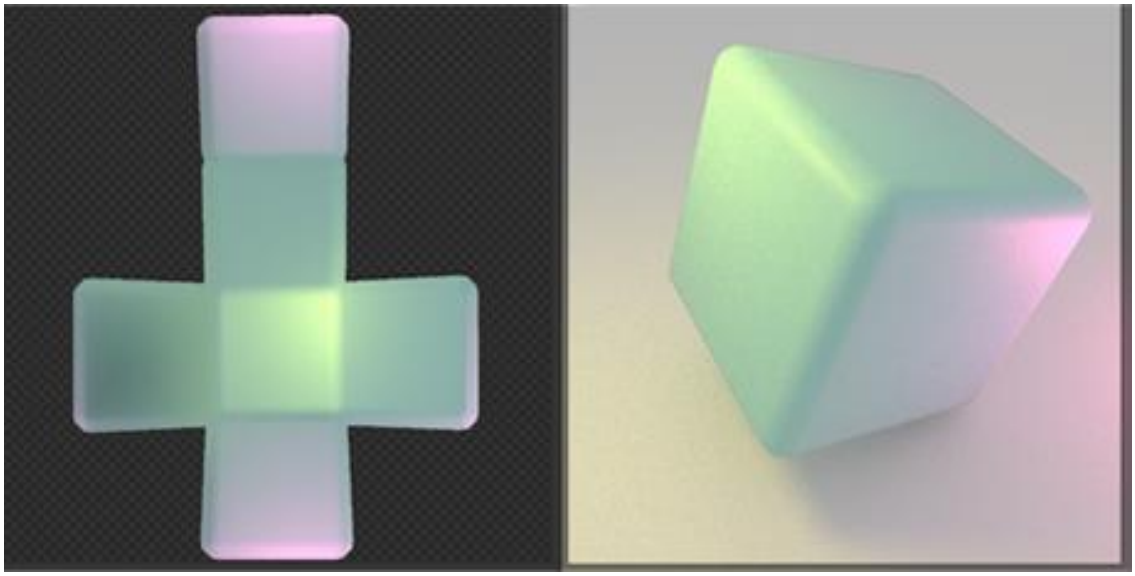
2.3 3D-grafiikka yleisesti ja tuotantolinjoissa

3D-grafiikalla tarkoitetaan yksinkertaisessa muodossa kolmiulotteisen geometrian esittämistä graafisessa muodossa, jossa mallinnetut asiat ja esineet muodostuvat pisteistä, joiden väliin on muodostettu tahkoja kolmioista, neliöistä ja ngoneista (pinta, jossa on enemmän kuin neljä pistettä). Mitä yksityiskohtaisempi malli on, sitä suurempi on sen sisältämien pisteiden ja tahkojen eli polygonien määrä. 3D-malleille voidaan määrittää materiaaleja, jotka määrittävät sen käyttäytymisen 3D-ympäristössä olevien elementtien kanssa. Näitä ovat muun muassa pistevalot, heijastuskartat ja taustavalaistus (ambient light). Materiaalien lisäksi iso osa 3D-mallien yksityiskohdista tulee erillisistä tekstuureista eli bittikartta-tiedostoista, joiden sisältämä pikselidata projisoidaan 3D-mallin pinnan päälle mallin 3D-tietoon sisältyvien UV-koordinaattien avulla.

Eri tekstuureilla voidaan muokata mallin materiaalin eri ominaisuuksia kuten heijastavuutta, diffuusiväriä, läpinäkyvyyttä ja pinnan normaalien suuntia. Materiaalien ominaisuudet riippuvat erillisestä shader-ohjelmasta, joka sisältää logiikan halutun grafiikan piirtämiselle. Shader-ohjelman ominaisuudet ja rajoitukset tulevat puolestaan alustan grafiikkamoottorista ja laitteistosta [6]. Ero mobiililaitteen ja pöytäkoneen grafiikkaprosessorien välillä on suuri, vaikkakin mobiililaitteiden suorituskyky on kehittynyt nopeasti.

3D-peleissä isoimmat tuotantolinjaan vaikuttavat tekijät ovat ennen kaikkea tekniset rajoitukset, joilla julkaisualustalla on suuri merkitys. Nykyajan keskivertoisella mobiililaitteella 3D-grafiikka toimii, mutta sen täytyy olla hyvin optimoitua parhaan mahdollisen suorituskyvyn aikaansaamiseksi. Dynaaminen valaistus ja normaalikarttojen käyttö täydessä 3D-pelissä on mahdollista, mutta se rajaa ison osan kohdelaitteista hyvän toimintakyvyn ulkopuolelle, mikä usein ei ole haluttua. Tässä yhteydessä hyvällä toimintakyvyllä tarkoitetaan noin 30:a näytönpäivitystä sekunnissa ilman suuria vaihteluita. Dynaamisten valaisujärjestelmien sijaan on suositeltavaa, että mahdollisimman paljon valon ja varjon käyttäytymiseen tarvittavaa laskentaa tehdään varsinaisen pelin

toiminnan ulkopuolella [7]. Tämä tarkoittaa valaistustiedon tallentamista tekstuurien muotoon, jolloin suurin osa tarvittavasta laskennasta on suoritettu etukäteen. Tämä mahdollistaa kevyempien shaderien käytön pelin malleissa silti toteuttaen lähes identtisen lopputuloksen. Esimerkiksi valaistustiedon tallentamista tekstuurin muotoon sanotaan ”leipomiseksi” (Baking), ja se on laajasti käytetty kaikkialla 3D-visualisoinnissa. Tästä on esimerkkinä kuvassa 1 näkyvä kuutio, jonka valaisutieto on tallennettu bittikuvaksi.



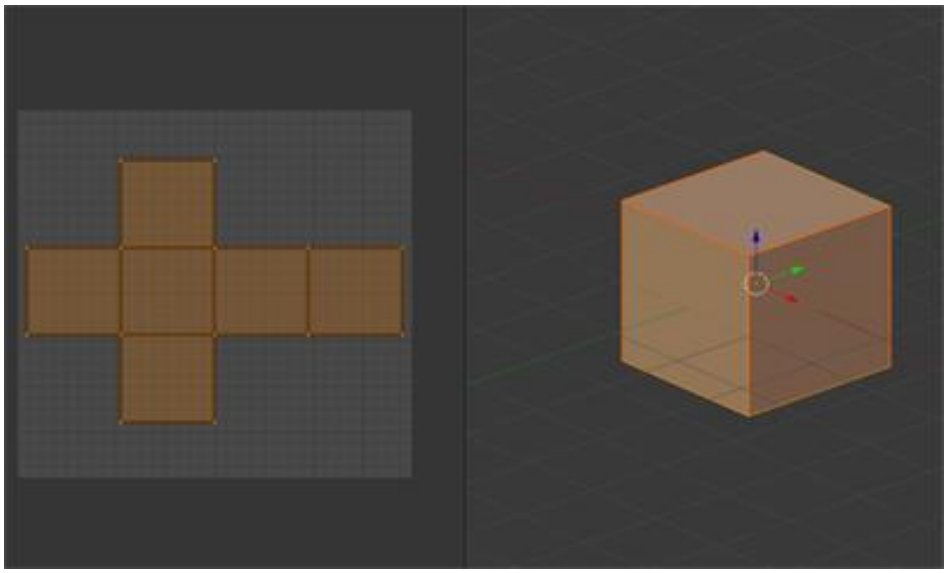
Kuva 1. Oikealla puolella olevan kuution valaisutieto on tallennettu vasemmalla olevaan kuvaan.

Tekstuurien käytössä tulee ottaa myös huomioon niiden koko ja formaatti. Kuten tietokonegrafiikassa yleensä, kaikkien tekstuurien ja kuvien resoluution tulee olla jokin kahden potenssi, kuten 32 x 32, 64 x 64, 512 x 512 ja niin edelleen. Usein ne voivat olla kuvasuhteeltaan 1:2, 1:4 tai 1:8, kunhan lopullinen resoluutio pysyy kahden potenssina [8]. Täytyy kuitenkin ottaa huomioon, että joillakin alustoilla ainoa tuettu kuvasuhde on 1:1. Tästä hyvänä esimerkkinä on laitevalmistaja Applen käyttämä pakkausmenetelmä PVRTC, joka tukee ainoastaan bittikarttatiedostoja, joiden kuvasuhde on 1:1. Sama pakkausmenetelmä rajoittaa myös resoluution pienimmillään 2 x 2 pikseliin. Mobiililaitteilla on yleisesti ottaen rajoitettu määrä välimuistia, mikä on hyvä muistaa tekstuurien kokoja päätettäessä. Esimerkiksi Applen Ipad 2 -laitteessa on 512 megatavua muistia, josta yksittäinen peli voi viedä enintään noin 200 megatavua. Jos peli vie liikaa välimuistia, ohjelma muuttuu epävakaaksi ja kaatuu helposti.

Vaikka pelin kaikkia tekstuureja ei ladattaisikaan laitteen välimuistiin, ne kasvattavat pelin kokoa. Tekstuurien koossa on myös hyvä ottaa huomioon niitä käyttävien mallien koko pelissä. Jos pelin malli on laitteen näytöllä vain muutaman kymmenen pikselin kokoinen, ei ole järkevää pitää siinä 512 x 512 pikselin kokoista tekstuuria, sillä suuri osa kaikesta kuvatiedosta menee hukkaan. Vaikka kaikki tekstuurit olisivat optimoituja ja tarkkaan säädettyjä kokonsa puolesta, ne voivat silti viedä huomattavan paljon resursseja. Tässä vaiheessa mahdollinen syy on pakkauksen puute tai sen väärä tyyppi. Bittikarttatiedostot sisältävät paljon tietoa, joka vie huomattavasti tilaa pakkaamattomana. 512 x 512 pikselin kokoinen tekstuuri vie yhden megatavun tilaa täysin pakkaamattomana 32-bittisenä alpha-kanavallisena kuvana, kun Android-laitteissa käytetty DTX5-formaatti vie vain puolet siitä. Pakkaaminen on olennainen osa tekstuurien kanssa työskentelyä, ja sen säätäminen oikein jokaiselle tiedostolle on edellytys onnistuneen tuotteen luomiseksi.

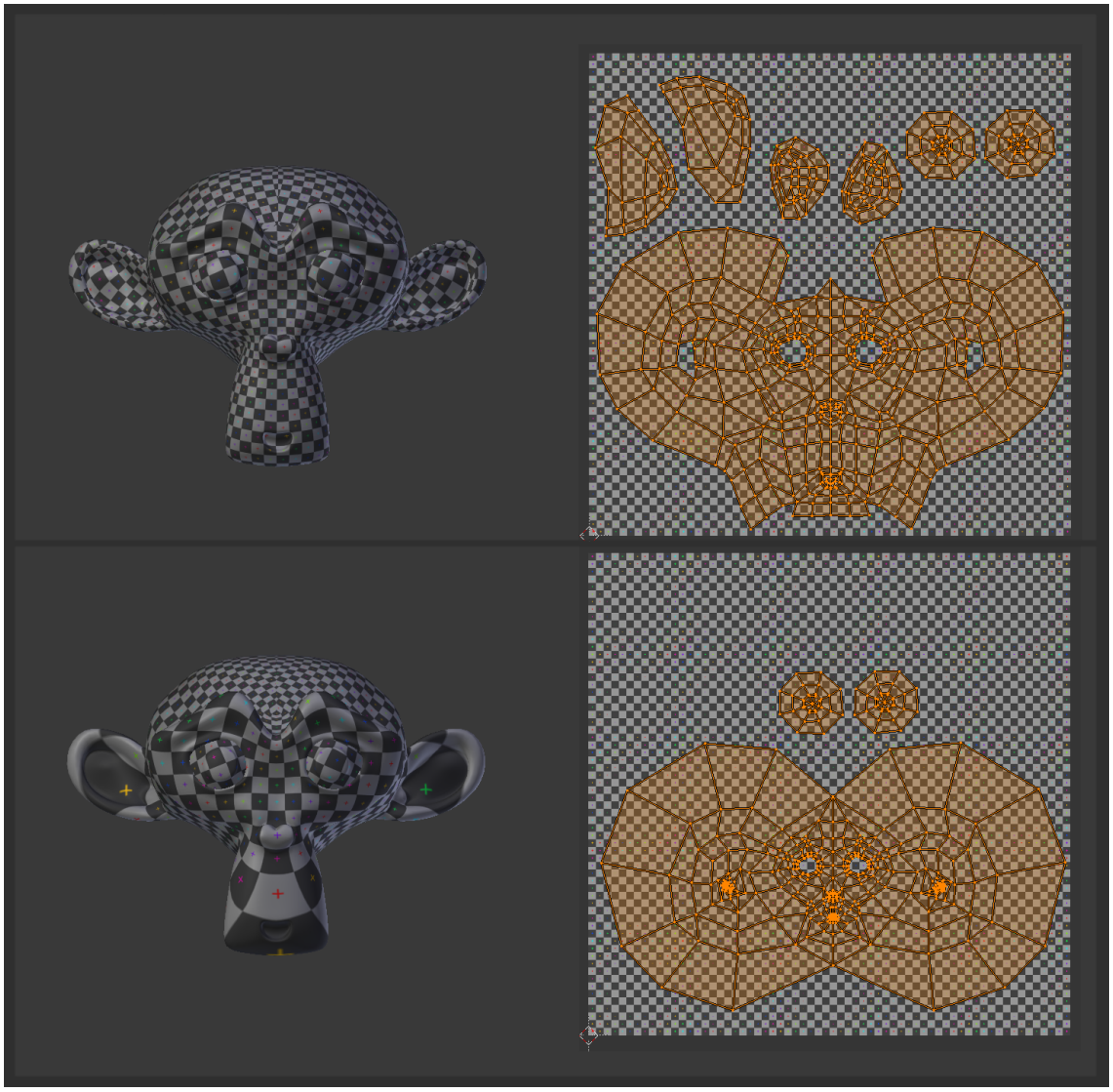
Kuten voi kuvitella, itse 3D-malleilla on myös suuri merkitys pelin ulkonäön ja toiminnan kannalta. Nykyaikaiset keskivertoiset mobiililaitteet pystyvät käsittelemään useita kymmeniä tuhansia polygoneja samanaikaisesti piirrettynä laitteen näytöllä, mutta vähätehoisemmat vanhemmat laitteet, kuten Iphone 3g, voivat käsitellä enintään noin kymmenen tuhatta polygonia kerralla [9]. Vaikka ylärajaa polygonien määrälle on mahdotonta määrittää, koska niiden vaikutus suoriutumiseen riippuu monesta tekijästä, on tärkeää mallintaa tarvittavat mallit mahdollisimman kevyiksi. Tämä tarkoittaa, että mallin tahkomäärä pysyy mahdollisimman alhaisena. Alhaisen määrän ja tarpeeksi korkean laadun tasapainottelu ei ole helppoa ja vaatii usein usean vuoden kokemuksen. Mallit näyttävät usein hyvinkin karkeilta sellaisinaan, mutta tämä ei ole ongelma, sillä niiden päälle projisoidaan yksi tai useampia tekstuureja, joiden avulla ”huijataan” malli näyttämään yksityiskohtaiselta.

Tekstuurien projisoiminen mallien päälle vaatii erillisten UV-koordinaattien määrittelyn; ne kertovat shader-ohjelmalle, kuinka ja missä tekstuuri näkyy mallin päällä. UV-koordinaattit määritetään usein 3D-mallinnusohjelmistossa, kuten Blender- tai 3DsMax-ohjelmistoissa, ja se tallennetaan 3D-mallin tietoihin. Nämä koordinaatit muodostavat UV-kartan, jota voidaan ajatella 3D-mallina avattuna kaksiulotteiselle tasolle, kuten kuvassa 2 näkyy. Sen toiminta on käsitteenä yksinkertainen. Jokaisen 3D-mallin tahkon sisälleen rajaama alue UV-kartassa vastaa samaa kohtaa itse tekstuurissa. Tämä tarkoittaa, että rajattu alue tekstuurissa näkyy mallin tahkossa sen rajaamalla tavalla. [10.]



Kuva 2. Blender-ohjelmistossa UV-kartoitettu kuutio.

UV-kartoituksella on suuri merkitys 3D-mallien valmistuksessa ja niiden kanssa työskentelyssä. Yleisesti ottaen UV-kartan tekemisen hankaluus on suoraan verrannollinen mallin monimutkaisuuteen ja yksityiskohtaisuuteen. Tämä on helposti huomattavissa monimutkaisissa hahmoissa tai malleissa, joissa on paljon sisäkkäisiä rakenteita tai monimutkaisia pinnanmuotoja. UV-kartoitusta ohjataan usein 3D-ohjelmistoissa löytyvillä UV-työkaluilla, joilla mallia saadaan avattua halutulla tavalla. Huonosti avattu malli kärsii usein suuresta määrästä venymiä UV-kartassa tai liian pienistä UV-saarekkeista, mikä on nähtävissä kuvassa 3 alemmassa osiossa. UV-kartoitus koetaan usein yhdeksi rasittavimmista askeleista 3D-mallin tekemisessä sen monimutkaisuuden takia, vaikka sitä varten kehitetyt työkalut ovat kehittyneet jatkuvasti.



Kuva 3. Huono UV-kartoitus venyttää mallin päällä olevia tekstuureja. Tämä on nähtävissä kuvan alemmassa mallissa.

UV-kartoituksen tärkeys korostuu etenkin pelikehityksessä, sillä resursseja on käytössä vain rajallinen määrä etenkin mobiililaitteilla. UV-kartan optimoinnin lähtökohta on mallin näkyvien alueiden priorisointi. Jos tiedetään, että esimerkiksi 3D-mallin pohjaa ei tulla näyttämään missään vaiheessa, on luonnollista antaa pohjassa oleville polygoneille vain hyvin vähän tilaa UV-kartassa. Tällä tavalla muut mallin osat saavat isomman osan tekstuuritilaa itselleen. Näkymästä poissa olevat polygonit voidaan myös poistaa mallista kokonaan, mikä voi olla olennaista tilanteissa, joissa poistettavien polygonien määrä on merkittävä suhteutettuna kohdelaitteen suorituskykyyn. Useissa pelimoottoreissa on olemassa toiminto, joka mahdollistaa ainoastaan kameran näkyvissä olevien polygonien piirtämisen (backface culling). Tämän toiminnon ansiosta mallin piilossa olevia polygoneja ei tarvitse poistaa. Täytyy kuitenkin ottaa huomioon, että polygonit

vievät silti tilaa välimuistista, vaikka niitä ei pelin ruudulle piirretä. Päätös polygonien poistamisesta tai säästämisestä riippuu monesta muuttujasta, kuten mallin tyypistä, resoluutiosta, käyttötarkoituksesta ja niin edelleen, ja kannattaa tehdä tapauskohtaisesti.

Toinen erinomainen tapa optimoida on UV-koordinaattien peilaus, jota usein käytetään malleihin, jotka ovat symmetrisiä vähintään yhden akselin suuntaisesti. Peilatun mallin UV-koordinaatit sijaitsevat toistensa päällä projisoiden saman osan tekstuuria kummallekin puoliskolle. Tällä tavalla tekstuureja voidaan pienentää jopa 25 prosenttiin niiden alkuperäisestä koosta. Päällekkäisiä UV-koordinaatteja voi hyödyntää myös malleissa, joissa on paljon samoja elementtejä. Esimerkkinä on 3D-malli, joka esittää koria täynnä omenoita. Jos omenoiden ei tarvitse olla erilaisia, ne voivat käyttää samaa kohtaa UV-kartassa. [11.]

Yleisesti ottaen 3D-pelin kehittäminen mobiililaitteille on erilaista kuin suorituskyvyltään huomattavasti paremmille tietokoneille teknisten rajoitusten takia. Tästä huolimatta niissä on hyvin paljon samaa, sillä ne perustuvat samaan tekniikkaan [12].

2.4 2D-grafiikka yleisesti ja tuotantolinjoissa

2D-grafiikalla tarkoitetaan kahdessa ulottuvuudessa esitettyä graafista kuvatietoa, joka muodostuu elementeistä. Näitä elementtejä ovat pääasiallisesti pikselit ja vektorit. Pikseliä voidaan kuvata pienimpänä ohjelmoitavana värielementtinä, joka on osa näyttöä tai kuvaa. Sanana ”pikseli” tulee englannin kielen sanoista picture element. Vaikka pikselit usein kuvataan ja toimivat neliöinä, ne voivat olla muodoltaan minkälaisia tahansa. Pikseli käsitteenä on abstrakti. Useimmissa värijärjestelmissä pikseli koostuu kolmesta tai neljästä kanavasta, joiden lopullinen arvo antaa sille sen lopullisen värin. Tästä on hyvänä esimerkkinä painotekniikassa käytetty CMYK-formaatti, jossa yhden painopisteen muodostavat keltainen, syaani, magenta ja harmaa kanava. Vektorigrafiikassa puolestaan elementteinä toimivat matemaattiset funktiot, jotka rajaavat alueita ruudulla muodostaen polygoneja tai tasoja. [13]

Vektori- ja pikseligrafiikka ovat erittäin käytettyjä formaatteja tietokonegrafiikan alalla, ja niillä on omat vahvuutensa ja heikkoutensa. Koska pikselikuvat koostuvat pienistä itsenäisistä elementeistä, niiden tarkka säätäminen ja hallitseminen ovat huomattavasti

vektorikuvia helpompaa. Tämän lisäksi niillä pystyy toistamaan huomattavasti enemmän värejä ja esimerkiksi valokuvia. Pikseligrafiikan eli rasterigrafiikan huonona puolena voidaan ajatella sen skaalautuvuutta [13]. Kuvan koko perustuu sen sisältämään pikselimäärään, mikä johtaa siihen, että kuvaa suurennettaessa sen yksityiskohdat sumentuvat, kuten kuva 4 osoittaa. Pienennettäessä kuvan yksityiskohtia taas katoaa. Vektorigrafiikassa puolestaan kuvan muodostavat matemaattiset kaavat säilyttävät kuvan yksityiskohdat täydellisesti skaalasta riippumatta. Tämän vuoksi vektorigrafiikalle ja rasterigrafiikalle on kehittynyt omat käyttötarkoituksensa. Esimerkiksi yritysten logot ja muut vastaavat useaa kokoa vaativat vähäsävytteiset graafiset elementit tehdään vektorigrafiikalla niiden skaalautuvuuden takia. Sama koskee myös useita painettavia tuotteita, kuten lehtiä, joiden ulkoasu on usein puhtaasti vektorigrafiikkaa. Rasterigrafiikan mahdollistamat yksityiskohdat ja värisyvyys ovat käytössä ennen kaikkea esimerkiksi valokuvissa, digitaalisissa maalauksissa ja teknisissä sovelluksissa. [14.]

Kuva 4. Vektorigrafiikalla tehdyt kuvat eivät menetä kuvatietoa koon muuttamisesta.

Peliteollisuudessa 2D-grafiikalla on ollut vakaa asema sen alkuvuosista asti. Ensimmäiset pelit kehitettiin jo 1950-luvulla, vaikka niitä ei tuolloin peleiksi varsinaisesti miellettykään. Nämä pelit olivat nykyaikaisiin peleihin verrattuna hyvin alkukantaisia, ja ne olivat käytännössä laboratorioinstrumenteista koottuja kokonaisuuksia. Tästä huolimatta esimerkiksi vuonna 1958 Willy Higinbothamin kehittämässä pöytätennispelissä pelin visuaalinen esitys toistettiin oskilloskoopista kaksiulotteisena animaationa. [15.]

Nykyään digitaalisella aikakaudella pelien 2D-grafiikka esitetään spriteinä, jotka ovat yksinkertaisuudessaan bittikarttakuvia, jotka edustavat visuaalisia elementtejä tai osia

niistä. Nykyajan spritejä voidaan kutsua ohjelmaspriteiksi, sillä ne muodostetaan ruudulle ohjelmiston kautta. Aikaisemmin, esimerkiksi 1980-luvulla markkinoille tulleessa Commodore 64 -tietokoneessa, oli erillinen laitteiston osa, joka muodosti pelin spritet näyttölaitteelle [16]. Spritejen käyttö peleissä ei ole koskaan loppunut, vaikka niiden suosio teollisuudessa hiipui 3D-grafiikan nousukauden aikana, joka alkoi 1990-luvun puolivälissä. Mobiilipeliteollisuuden myötä 2D-grafiikka ja spritejen käyttö on lähtenyt uuteen nousuun.

2D-grafiikaa käytetään peleissä yleensä sen keveyden, tyylin ja helppokäyttöisyyden takia. Toisin kuin 3D-grafiikassa, spriten esittäminen pelissä ei vaadi erillistä 3D-mallia, jonka ympärille kuva käärittäisiin [17]. Modernit 2D-moottorit, kuten Unity 3D ja Ogre3D, generoivat kuvan ympärille 3D-mallin automaattisesti. 3D-mallin avulla elementtiä on helpompi manipuloida ja renderöidä. Useimmat 2D-moottorit ovat käytännössä samanaikaisesti 3D-moottoreita, joten periaatteessa tällaisia kehitysympäristöjä voidaan kutsua 2,5D-moottoreiksi tai editoreiksi. [18.]

Toisin kuin 3D-pelien kehityksessä, 2D-spritejen valmistus ja käyttöönotto on 2D-malleja suoraviivaisempaa. Vaikka 2D-spritet ovat käytännössä 3D-objekteja, niiden kanssa yleisesti käytetyt shader-ohjelmat eivät vaadi dynaamisia valonlähteitä toimiakseen oikein eivätkä erillisiä tekstuureja ohjaamaan valon interaktiota objektin pinnan kanssa. Tämä takaa samalla sen, että sprite näyttää pelissä samalta, kuin miltä se näytti editointiohjelmassa. Spritejen valmistamiseen käytetään 2D-editointiohjelmia, kuten Adoben laajasti käytettyä Photoshopia, jotka pystyvät tuottamaan tarvittavan hyvää laatua ja tallentamaan tuotetut spritet pelimoottorille sopivassa muodossa.

Yleisesti ottaen 2D- tai 2.5D-pelissä käytettävät spritet ja muut elementit eivät vaadi toimiakseen mitään muuta kuin bittikarttatiedoston, siinä missä 3D-peli voi vaatia 3D-mallin lisäksi enemmän kuin neljä erillistä tekstuuria, jotka ohjaavat mallille annetun materiaalin ominaisuuksia kuten heijastavuutta, diffuusioväriä ja pinnan normaalien suuntaa. Tällä perusteella voidaan sanoa, että 2D-pelien kehittäminen on helpompaa kuin 3D-pelien. On kuitenkin selvää, että lähestyttäessä erityyppisiä pelejä tapauskohtaisesti, joissakin 2D-peleissä on käytetty huomattavasti ”normista poikkeavia” menetelmiä, jotka nostavat sen kehityksen vaikeuden huomattavasti keskivertoista 3D-peliä korkeammaksi.

Usein 2D-editointiohjelmisto on ainoa työkalu, jota spritejen valmistamiseen tarvitaan, mutta ajoittain muunkin tyyppisistä ohjelmistoista on paljon hyötyä. 3DsMaxin ja Blenderin tapaiset ohjelmistot sopeutuvat hyvin animaatioiden ja visuaalisten efektien tekoon. Animaatiot ja efektit renderöidään ohjelmasta yksittäisinä kuvina, jotka jälkeenpäin syötetään pelimoottoriin sellaisinaan tai kuva-arkkina (spritesheet) [19]. Kun monimutkaiset animaatiot ja visuaaliset efektit renderöidään ulkoisessa ohjelmassa spriteiksi, pelin resursseja säästyy huomattavasti, sillä monimutkaista laskentaa ei tarvitse suorittaa pelin toiminnan aikana. Tätä tekniikkaa käytetään myös vaativissa 3D-peleissä sen tehokkuuden takia. Ajoittain pelin toiminnan aikana generoitu animaatio saattaa olla kuvasarja-animaatiota parempi vaihtoehto suorituskyvyn kannalta, jos kiinteässä muistissa tai välimuistissa ei ole tilaa suurille määrille kuvadataa.

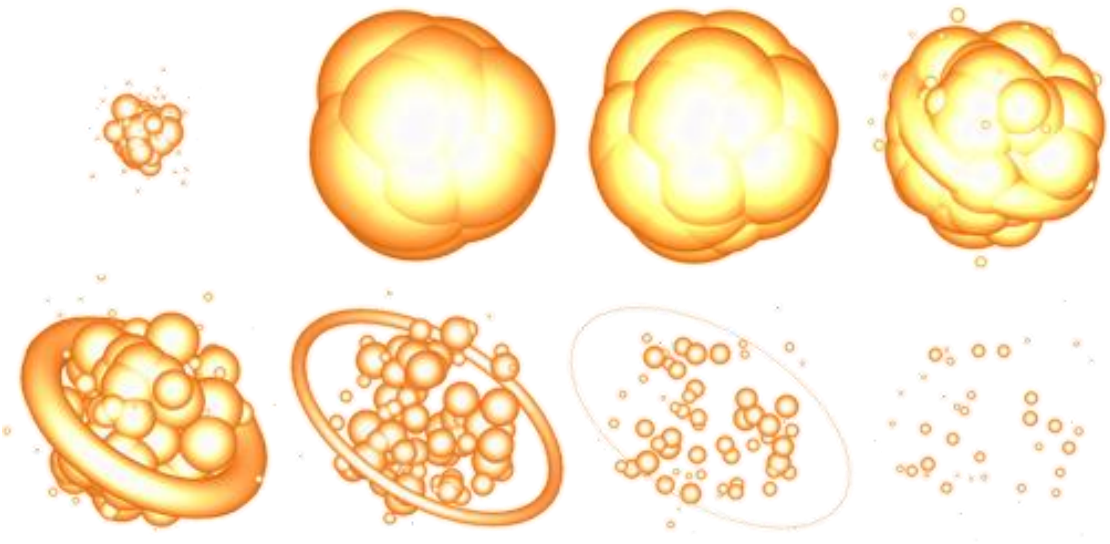
Photoshopin kaltaisten ohjelmien lisäksi laitevalmistajakohtaiset optimointityökalut saattavat tarjota helpotusta kehityksessä etenkin mobiililaitteille. Hyvänä esimerkkinä ovat laitevalmistaja Applen käyttämään PVRTC-pakkausformaattiin perustuvat tekstuuriesikatselutyökalut. PVRTC-formaatin pakkausta ei voida esittää visuaalisesti esimerkiksi Unity 3D-pelimoottorissa, mikä hankaloittaa huomattavasti tekstuurien ja spritejen pakkauksen säätämistä halutulle tasolle. Valitettavasti nämä työkalut vaativat Applen valmistaman tietokoneen tai laitteen. [20.]

Nykypäivänä perinteisille tietokoneille kehitettäessä 2D-pelien tekniset rajoitukset ovat suhteellisen vähäiset, sillä laitteistot ovat tulleet vuosi vuodelta tehokkaammiksi. Samaa ei voida sanoa kehitettäessä pelejä mobiililaitteille. Suurin ongelma mobiililaitteille kehittämisessä on käytössä olevien laitteiden kirjo. Vaikka uusimmat mobiililaitteet ovat tehokkaita, niiden osuus kaikista käytössä olevista laitteista on hyvin pieni. Jos peli, jota ollaan kehittämässä, on kaupallinen projekti, on järkevintä rakentaa peli siten, että se toimii mahdollisimman monella laitteella.

Suurimmat ongelmat keskivertoisella mobiililaitteella muodostuvat pelin renderöimien spritejen määrästä. Jokainen kuvallinen objekti laitteen näytöllä muodostaa yhden piirtopyynnön (drawcall) laitteen suorittimelle. Piirtopyyntö on pelin lähettämä pyyntö laitteen grafiikkarajapinnalle. Laitteen suorittava komponentti joutuu tekemään huomattavasti työtä jokaista pyyntöä kohden, minkä takia niiden liiallinen määrä aiheuttaa helposti pullonkaulan syntymisen suorittavalle komponentille, mikä puolestaan aiheuttaa kehysnopeuden laskemisen [21]. Tämä näkyy pelissä nykimisenä. Piirtopyyntöjen suositeltu määrä on laitekohtaista, ja siihen vaikuttaa myös itse pelimoottori ja suorittavan

komponentin yleinen kuormitus. Vaikka piirtopyyntöjen määrä olisi huomattavasti suositeltua määrää korkeampi, siitä ei tarvitse välittää, jos kehysnopeus on silti hyväksyttävissä lukemissa.

Piirtopyyntöjen määrää voi vähentää käyttämällä samaa materiaalia mahdollisimman monessa spritessa. Tällöin useampien spritejen piirtäminen voidaan suorittaa yhdellä pyynnöllä useampien sijaan, mikä usein vähentää tarvittavien piirtopyyntöjen määrää huomattavasti. Tämän lisäksi pelissä käytetyt spritet voidaan koota yhteen kuvatiedostoon, jota kutsutaan atlakseksi, jota kuva 5 esittää. Atlasointi voidaan suorittaa manuaalisesti pakkaamalla pelin spritet yhteen suureen kuvatiedostoon esimerkiksi Photoshopin avulla tai automaattisesti atlasten tekoon erikoistuneilla ohjelmilla tai pelimoottorin tarjoamilla työkaluilla. Kun sprite-atlas on tehty, jokaiselle spritelle määritetään kohta, josta se ottaa kuvatietonsa. Jos atlas on generoitu esimerkiksi Unity-pelimoottorilla, spritejen koordinaatit atlaksessa määrittyvät automaattisesti. Kaikkien atlakseen pakattujen kuvien kompressio ja kuvaformaattiin vaikuttavat tekijät ovat samat, sillä atlas käsitellään yhtenä kuvana.



Kuva 5. Kahdeksan kehyksen kuva-atlas räjähdysanimaatiolle.

Toinen huomioonotettava tekijä pelin toiminnassa mobiililaitteilla on laitteen niin sanottu täyttönopeus (fillrate). Täyttönopeudella tarkoitetaan grafiikkaprosessorin kykyä laskea pikseleitä ruudulla jokaista sekuntia kohti. Täyttönopeus on laitesidonnainen, ja se on vakio. Shaderit, jotka käyttävät läpinäkyvyyttä ja sekoitustiloja, ovat usein raskaita täyttönopeuden kannalta, sillä läpinäkyvät pikselit joudutaan laskemaan useaan kertaan.

Tähän on syynä se, että jokaisen läpinäkyvän pikselin takana olevat pikselit joudutaan ottamaan mukaan shaderin sisällä tapahtuviin laskuihin, jotta lopputuloksena piirretyn pikselin väri olisi oikea. Läpinäkyvyyttä voidaan käyttää maltillisesti, kunhan ruudulla piirrettävien pikselien määrä ei nouse liian korkeaksi. Liian korkeaksi noussut pikselimäärä pidentää huomattavasti kehysaikaa laskien kehysnopeutta ja tehden pelistä pahimmillaan käyttökelvottoman.

3 Pelin suunnitelma tuotantolinjan pohjana

3.1 Pelidesign määrittävänä tekijänä peleissä

Vaikka pelit eroavat perinteisistä palveluista ja tuotteista siinä, että niitä ei varsinaisesti ole luotu ratkaisemaan jotain tiettyä ongelmaa, on niiden lähtökohta silti sama. Pelit perustuvat aina johonkin ydinideaan ja suunnitelmaan, jonka mukaan peli toimii [22]. Suunnitelma, joka määrittää pelin toimintaa, sen mekaniikkoja, tuntumaa ja tunnelmaa, on pelin perustus, jonka päälle se rakennetaan. Tästä syystä pelin suunnitelmaksi kutsuttu GDD (Game Design Document) on jokaisen peliprojektin sydän ja samalla salaisin asiakirja.

Pelisuunnittelu on alana suhteellisen nuori, ja se kehittyy nopeasti. Sitä voidaan pitää usean eri alan risteytymänä, sillä hyvän pelin suunnitteleminen vaatii usein sen jokaisen osa-alueen ymmärtämistä. Etenkin ohjelmointitaidoista on huomattavan paljon etua, sillä se mahdollistaa erilaisten prototyyppien rakentamisen ja auttaa hahmottamaan, kuinka paljon resursseja erilaisten mekaniikkojen käyttöönotto veisi. Prototyyppien tekeminen on keskeistä kokeellisten mekaniikkojen testauksessa. Uusi innovatiivinen pelimekaniikka saattaa vaikuttaa kirjallisessa muodossa erittäin lupaavalta, mutta käytännössä se saattaa olla käyttökelvoton epäintuitiivisuuden tai muun seikan vuoksi. [23.]

Ohjelmointitaitojen lisäksi graafisesta suunnittelutaidosta ja äänisuunnittelun osaamisesta on huomattavasti hyötyä. Visuaalisella ulkonäöllä voi olla hyvin suuri vaikutus pelin tunnelmaan, kiinnostavuuteen ja lopulliseen toimivuuteen, minkä vuoksi sen hyväksi käyttäminen peliä suunnitellessa on olennaista. Tarkoituksena ei ole suunnitella pelin graafista ilmettä alusta loppuun, vaan asettaa sille raja-arvoja, joiden pohjalta ryhmän graafinen suunnittelija voi aloittaa pelin graafisen ilmeen työstämisen. Graafisen tyylin ja elementtien määrittelyn tarkkuus riippuu kuitenkin siitä, minkälaista peliä ollaan suunnittelemassa. Hyvänä esimerkkinä tästä ovat tekstiseikkailupelit, joissa ei välttämättä käytetä juuri ollenkaan graafista ulkoasua. Päinvastaisena esimerkkinä voidaan pitää vuonna 1998 julkaistua Fallout 2:ta, joka on kuvassa 6 näkyvä post-apokalyptiseen maailmaan sijoittuva vuoropohjainen, taktiseen pelaamiseen keskittynyt seikkailupeli. Sen alkuperäisessä suunnitelmassa oli määritelty, että pelin visuaalisen

ulkonäön tulee vastata ydinsodan jälkeistä julmaa ja väkivaltaista maailmaa. Graafisen tyylin ja elementtien määrittely ei ole aina olennaista tehdä, jos pelin visio ei sitä vaadi.

Kuva 6. Fallout 2:n graafinen tyyli sopii hyvin post-apokalyptiseen maailmaan [24].

Pelisuunnittelussa on tärkeää tuntea eri kategorioihin kuuluvien pelien toimintaa myös käytännön tasolla [25]. Tämä tarkoittaa pelien pelaamista. On tärkeää ymmärtää, miten pelit toimivat ja ovat toimineet aikaisemmin, jotta suunnittelijalla on helpompaa vastata seuraaviin kysymyksiin:

- Mitä on aikaisemmin tehty?
- Minkälaiset pelit toimivat?
- Minkälaiset pelit ovat tällä hetkellä nousussa?
- Mitä ei ole vielä tehty?
- Minkälainen visuaalinen tyyli sopii tiettyihin peleihin?
- Minkälainen kilpailu peligenrejen sisällä on?

- Miten erilaiset mekaniikat on käyttöönotettu erilaisissa peleissä?

Oman alan tuotteiden ymmärtäminen auttaa keksimään uusia tapoja lähestyä erilaisia ongelmia ja ratkaista niitä innovatiivisilla tavoilla sekä kartoittaa, minkälaiset pelit menestyvät ja minkä takia. Pienetkin erot pelattavuudessa ja käyttäjäkokemuksessa voivat erottaa pelin kaikista muista vastaavista peleistä. Uusia pelejä julkaistaan yhä nopeammin, mikä kasvattaa kilpailua jatkuvasti. Vaikka visuaalinen ulkonäkö on yksi suurimmista vaikuttajista uusien pelaajien hankinnassa, se ei anna pelaajalle syytä pelata peliä pitkään. Hyvin suunniteltu peli pitää pelaajat pelissä pidemmän aikaa ja antavat heille syyn tulla takaisin. Tämä on erityisen tärkeää Free-to-play-ansaintalogiikkaa käyttävissä peleissä, joissa pelaaminen on ilmaista, mutta tuotot tulevat mainoksien tai mikromaksujen kautta. Näiden tuottojen määrä on suoraan verrannollinen pelaajapopulaation suuruuteen. [26, s. 14.]

Pelisuunnittelu on työnä vapaata, ja niin sanottuja oikeita tapoja suunnitella pelejä ei ole olemassa. Erilaisia lähestymistapoja pelien suunnittelemiseen on olemassa, ja niitä usein sovelletaan tilanteesta riippuen. Jokaisella pelisuunnittelijalla on usein omat toimintatapansa lähestyä erilaisia suunnittelussa syntyneitä ongelmia ja soveltaa aikaisemmin opittua tietoa peleistä.

3.2 Tarvittavien kehitysaskelten erittely

Koska pelisuunnitelma on pelin perustus, on olennaista, että siinä käytettävät pelielementit vastaavat suunnitelmassa määrättyjä tarpeita. Tästä syystä tuotantolinjojen rakentamisessa tulee ottaa huomioon suunnitelmassa määritetyt vaatimukset tarkasti. Tilanteessa, jossa vaatimuksia ei ole määritelty tarkasti tietylle elementtityypille, voidaan niiden kanssa työskennellä suhteellisen vapaasti, mutta niiden tulee silti noudattaa kohdelaitteen teknisiä rajoituksia.

Tuotantolinjaa rakennettaessa esimerkiksi visuaalisille elementeille, kuten spriteille ja 3D-malleille, kaikki tuotantolinjaan liittyvät alustavat päätökset perustuvat pelin suunnitelmaan ja aluksi ennen kaikkea valittuun alustaan, jolle peli tullaan julkaisemaan. Julkaisualusta on etenkin visuaalisten elementtien lisäksi erittäin suuri tekijä, sillä esimerkiksi mobiililaitteet ovat suorituskyvyltään huomattavasti pöytätietokoneita heikompia ja muistikapasiteetiltaan pienempiä. Tämä tekijä yksinään määrittää, kuinka monimutkaisia 3D-mallit voivat olla tai kuinka suuria tekstuureja tai spritejä pelissä voidaan käyt-

tää. Mobiililaitteiden pienet näytöt vaikuttavat myös elementtien suunnittelemiseen, sillä niiden tulee olla selkeästi hahmotettavissa kaikissa tilanteissa.

Julkaisualustan jälkeen seuraava tuotantolinjaa määrittelevä tekijä on itse pelin suunnitelma, joka määrittelee, minkälainen peli on kokonaisuudessaan. Visuaalisten elementtien kannalta merkittävimpiä määritelmiä ovat seuraavat:

- 2D- vai 3D-pohjainen grafiikka
- yleinen graafinen tyyli
- animaatioiden ja tehosteiden käyttö
- interaktio pelimekaniikan kanssa
- graafisten elementtien määrä ruudulla
- tarvittavien elementtien kokonaismäärä
- pelin mittakaava
- käytetty kamerakulma.

Yleisesti 2D- ja 3D-pelien tuotantolinjat eroavat merkittävästi toisistaan, sillä niiden valmistukseen tarvittavat työvaiheet ovat erilaiset. Suunnitelmassa määritetty yleinen graafinen tyyli voi vaikuttaa valittuun grafiikkatyyppiin, sillä esimerkiksi piirrosmaisen visuaalisen ulkonäön saaminen pelille on usein helpointa käsin piirretyillä 2D-spriteillä tai rasteroidulla vektorigrafiikalla. 3D-grafiikkatyyppi voidaan valita sen mahdollistaman estetiikan ja joustavuuden ansiosta.

2D- ja 3D-pelien graafisten elementtien tuotantolinjaa suunniteltaessa ensimmäiseksi askeleeksi asetetaan konseptointivaihe, jossa muodostetaan selkeä suunnitelma valmistettavasta elementistä. Konseptitaiteen tekee usein elementin kanssa työskentelevä artisti tai erillinen konseptiartisti riippuen pelin kanssa työskentelevästä ryhmästä. Konseptitaiteella on erilaisia vaatimuksia riippuen siitä, minkälaista peliä ollaan tekemässä, minkälaista pelielementtiä konsepti kuvaa ja tullaanko sitä käyttämään ainoastaan ryhmän sisäisesti. Yleisesti ottaen konseptikuvan tulee välittää selkeä kuva siitä, miltä elementti tulee näyttämään eri suunnista ja miten se toimii. Sen ei tarvitse olla viimeistelty, sillä sitä ei sellaisenaan käytetä pelissä. Hyvän konseptikuvan tulee olla yhtä selkeä muille ihmisille kuin sen tekijälle, sillä joku toinen saattaa joutua työskentelemään

sen kanssa myöhemmin. Konseptointivaihe ei ole välttämättä oleellinen vaihe tuotannossa pelille, joka sisältää vähän grafiikkaa.

Konseptitaiteen merkitys korostuu etenkin 3D-assetteja tehtäessä, sillä mallien rakenne määrittää niiden toimivuuden esimerkiksi animaatioiden kanssa. Tämän vuoksi konseptitaiteen täytyy välittää selkeä kuva siitä, miten malli tulee toimimaan pelissä ja minkälaisia funktioita sillä tulee olemaan. Kaikki mallin rakenteelliset ominaisuudet tulee määritellä konseptitaiteessa, jotta niitä ei tarvitse tehdä mallintamisen aikana. Mallille suunniteltujen rakenteiden tietäminen etukäteen auttaa 3D-artistia rakentamaan mallin paremmin ja nopeammin.

Ennen graafisen tuotannon alkua on syytä päättää graafista tyyliä ja elementtejä koskevat tavoitteet, määritteet ja raja-arvot. Tätä graafista ohjeistoa kutsutaan teollisuudessa usein ”art bibleksi”, joka koostuu esimerkeistä halutusta lopputuloksesta, viittauksista, hyväksyttyjen värien ja muotojen listoista ynnä muuta sellaista. Sen tarkoitus on pitää tuotettujen elementtien laatu johdonmukaisena ja hallittuna, ja se auttaa koko työskentelevää ryhmää ymmärtämään, mihin suuntaan pelin taidesuuntaus on menossa. Graafisen ohjeiston rakentaminen on iteratiivinen prosessi, mutta se kannattaa tehdä valmiiksi ennen elementtien tuotantoa. Muutokset ovat mahdollisia, mutta niitä täytyy tehdä harkiten. [27.]

Kun elementin konsepti on valmis, sen valmistaminen voidaan aloittaa. Elementin luominen 3D- tai 2D-ohjelmistossa on yleisesti ottaen suurin työvaihe tuotantolinjassa ja samalla tärkein lopputuloksen kannalta. Tässä työvaiheessa tapahtuvat prosessit riippuvat ennen kaikkea pelin suunnitelmasta, graafisesta ohjeistosta, julkaisualustasta ja sen tuomista rajoitteista. Tämän vuoksi tapoja toteuttaa kyseinen vaihe on lukuisia.

Julkaisualustan ja valitun pelimoottorin yhteenlasketut rajoitteet toimivat pohjana tuotantovaiheen prosessien määrittelemiselle. Esimerkiksi mobiililaitteille tehtävässä pelissä ei voida käyttää korkean resoluution malleja, suuria tekstuureja tai moderneja materiaaleja, kun taas pöytätietokoneelle kehittäminen on huomattavasti vapaampaa. Julkaisualusta vaikuttaa enimmäkseen mallien ja spritejen tarkkuuteen ja kokoon sekä niiden riippuvuuteen ennalta lasketusta tiedosta. Tästä hyvänä esimerkkinä toimii korkearesoluutioisesta 3D-mallista tallennettu kuvatieto, jota käytetään komponenttina alhaisen resoluution 3D-mallin tekstuureissa.

Pelin suunnitelmalla on suuri vaikutus graafisten pelielementtien luomisessa. Haluttu tyyli, muotokieli ja kunkin objektin toiminnallisuudet vaikuttavat merkittävästi mallin tai spriten elementtien rakenteeseen. Valmistusvaiheessa käytetyissä ohjelmissa eri toiminnoille voidaan määrittää arvoja ja vaatimuksia, joiden mukaan elementti tulee valmistaa. Tämä voi käytännössä tarkoittaa tiettyjen muokkaajien (modifier) käyttöä, tasohierarkioiden rakennetta, tiettyjen toimintatapojen käyttöä, automatisointia, suodattamista ja muuta vastaavaa. Kaikkia tapauksia on mahdotonta määritellä, mutta periaatteena toimii tasalaatuisen lopputuloksen saaminen. Tuotantolinjalle päätetyt määritteet eivät ole absoluuttisia, sillä samat määritteet eivät välttämättä toimi kaikille elementeille. Tämän vuoksi määritteitä joudutaan ajoittain soveltamaan. Lopputuloksen tulee kuitenkin olla yhteensopiva muiden linjalta valmistuneiden pelielementtejen kanssa.

Valmistusvaihe voi koostua monesta eri vaiheesta. 3D-mallien kanssa valmistukseen kuuluu mallinnus halutussa ohjelmassa (3DsMax, Blender jne.), teksturointi kuvankäsittelyohjelmassa tai 3D-ohjelmistossa 3D-maalauksen avulla. 3D-mallien animointi voidaan tehdä 3D-ohjelmistoissa mallintamisen ja teksturoinnin jälkeen erillisessä vaiheessa. Tämä on yleistä etenkin silloin, jos animoinnin tekee joku muu kuin itse assetin mallintaja ja peli sisältää paljon animaatioita. Vaikka pelissä ei olisikaan montaa animaatioita, on järkevää tehdä ne erikseen, sillä niiden muokkaaminen ja säätäminen ovat tällöin helpompaa.

Useiden 2D-assettien valmistusvaihe koostuu pelielementtien luomisesta valitussa ohjelmistossa ja jälkikäsittelystä. Digitaalisesti käsin piirretyt spritet ja muut graafiset elementit valmistetaan usein kuvankäsittelyohjelmistoissa, joissa määrittäviksi tekijöiksi määritetään esimerkiksi resoluutio, kuvasuhde, suodatus, sivellinasetukset ja niin edelleen [28]. 3D-malleista renderöityjen 2D-kuvien valmistus puolestaan sisältää paljon yhteneväisyyksiä 3D-assettien valmistuksesta, sillä suuri osa työstä tehdään 3D-ohjelmistossa. Renderöidyn kuvan lopulliseen muotoon saattaminen ja yhtenäistäminen vaativat usein jälkikäsittelyä. Tähän voidaan käyttää kuvankäsittelyohjelmistoja tai videonkäsittelyohjelmistoja, kuten Adoben After Effects -ohjelmistoa. Tulosten yhtenäistäminen vaatii usein samojen säätötasojen käyttöä kaikille elementeille. Tähän kuuluvat muun muassa värikorjaus ja erilaiset visuaaliset tehosteet, joita on vaikea toteuttaa 3D-ohjelmistossa. Samalla tavalla kuin muidenkin graafisten elementtien valmistuksessa, määritteitä voi soveltaa, kunhan elementtien yhtenäisyys säilyy.

3.3 Optimointi ja ennakointi

Tuotantolinjaa määriteltäessä on tärkeää huomioida tuotannon tehokkuus. Eri vaiheiden viemä aika täytyy punnita niiden antaman lopputuloksen mukaan. Jos todetaan, että vaiheen tekemiseen menee arviolta kaksi tuntia ja sen vaikutus lopputulokseen on marginaalinen, on olennaista jättää se pois tuotantolinjasta. Tuotantolinjan monimutkaisuuteen ja sen karsimiseen vaikuttavat etenkin käytettävissä olevat resurssit, kuten aika, käytettävän työvoiman määrä ja lopullisen tuotteen tuotantoarvot. Lisäksi tuotantolinjan läpi kulkevien elementtien määrällä on suuri vaikutus, etenkin jos minimitekijänä toimivat työvoiman riittämättömyys tai ajan puute. Tämä voidaan ratkaista hyväksyttävän laadun laskemisella tai ulkoistuksella. [29.]

Pelielementtien tuottaminen sisältää usein paljon manuaalista toistuvaa työtä, kuten tiedostojen koon muuttamista, elementtien tallentamista ja eri arvojen muuttamista. Toistuva manuaalinen työ vie paljon aikaa, sen suorittaminen on tehotonta ja virhealtista ja sen tekeminen ei yleisesti ottaen ole mielekästä työntekijälle. Työskentelemiseen käytetty aika on järkevää kuluttaa tekijän luovaa kykyä vaativiin työvaiheisiin. Tästä syystä tuotantolinjan osien automatisointi on tärkeää. Monet ohjelmat, kuten Adoben Photoshop, tukevat monipuolista automatisointityökalustoa, joista osa ei vaadi ohjelmointiosaamista. Automatisointityökaluja hyödyntämällä tuotantolinjan tehokkuus nousee, virheiden määrä laskee ja työ pysyy mielekkäänä.

Tapauksessa, jossa automatisointityökaluja tai muita tuotantolinjaa nopeuttavia tekijöitä ei ole ohjelmistoissa valmiiksi saatavilla, on mahdollista löytää kolmannen osapuolen tarjoamia lisäosia, jotka tekevät halutun työvaiheen helpommaksi. Lisäosien maksullisuus riippuu usein käytettävästä ohjelmasta. Vaikka lisäosa maksaisikin, on tärkeää arvioida sen hinta verrattuna säästettyyn aikaan, etenkin tilanteessa, jossa työtekijälle maksetaan palkkaa.

Tuotantolinjan rakenne pysyy harvoin muuttumattomana koko projektin ajan, sillä odottamattomia muutoksia tapahtuu usein pelinkehityksessä. Tästä syystä on tärkeää rakentaa tuotantolinja siten, että se pysyy helposti muokattavissa. Käytännössä muokattavuutta voidaan lisätä käyttämällä versionhallintaa ja selkeää tiedostorakennetta, jossa versiot, valmiit tuotokset ja työtiedostot säilytetään erikseen. Tuotantolinjassa käytettyjen ohjelmien määrä kannattaa pitää mahdollisimman vähäisenä ja tallennettujen tiedostojen formaatit avoimina, jos vain mahdollista.

4 Vulpine Games -pelistudio ja Last Planets -peli

4.1 Vulpine Games yrityksenä

Vuonna 2014 perustettu Vulpine Games Oy on pelien ja median tuotantoon keskittynyt yritys, joka työllistää kuusi henkilöä. Se on virallisesti julkaissut kaksi pienen mittakaavan mobiilipeliä nimeltään Color Square ja Hipster Blender, jotka ovat saatavilla Android-käyttöjärjestelmää käyttäville puhelimille ja Windows-puhelimille. Vaikka yritys virallisesti rekisteröitiin vuonna 2014, se oli aktiivinen jo ennen sitä. Vuonna 2013 Metropolia-ammattikorkeakoulun järjestämällä innovaatioprojektikurssilla muodostettu ryhmä päätti jatkaa aloittamaansa projektia kurssin loputtua. Kaikki ryhmässä olleet henkilöt eivät lähteneet mukaan pelin eteenpäin kehittämiseen.

Innovaatioprojektista alkunsa saanut Last Planets -peli oli alun perin mobiilipeliksi suunnattu arcade-tyyppinen planeetan puolustukseen keskittynyt 2D-peli, joka oli suunnattu yli seitsemänvuotiaille. Sen pelaamisen ydinidea oli puolustaa ruudun keskellä olevaa planeettaa ruudun sivulta tulevilta asteroideilta, kuten voi nähdä kuvassa 7. Planeetan puolustaminen tapahtui painamalla ruutua, jolloin planeetta ampui ammuksen planeetan ytimen ja painalluskohdan välille muodostuneen suoran mukaisesti. Kehityksen edetessä uusia pelikenttiä, ammuksia ja muita ominaisuuksia lisättiin tuomaan peliin lisää sisältöä, ja se käännettiin aikaisemmasta kehitysympäristöstä Unity-pelimoottorille. Pelin ansaintamalli oli Free-to-Play, ja mikromaksuilla oli mahdollista ostaa erilaisia Power up -lisävoimia, joiden avulla kenttien läpipeluu helpottuisi. Peliä kehitettiin yli vuoden verran, ja se oli suhteellisen valmis julkaisuun vuoden 2014 lopussa. Peli ei kuitenkaan koskaan päässyt kuluttajamarkkinoille, sillä sen kehitys lopetettiin alhaisen kannattavuuden vuoksi.

Kuva 7. Last Planets -pelin ensimmäinen versio.

4.2 Last Planets -peli

Vanhan Last Planets -pelin tilalle käynnistettiin uusi projekti, joka nimettiin samalla nimellä. Se on huomattavasti edeltäjäänsä haastavampi peli kehityksen kannalta, ja se perustuu aikaisemmin toimiviksi todistettuihin malleihin. Uusi Last Planets on mobiililaitteille suunnattu (ios / Android) free-to-play massiivinen moninpeli. Peli on käytännössä tukikohdanpuolustuspeli, jossa jokaisella pelaajalla on oma planeetta, jota täytyy suojella erilaisilla puolustavilla rakennuksilla muiden pelaajien hyökkäyksiä vastaan. Toisin kuin pelin vanhemmassa versiossa, puolustaminen on passiivista eli siihen ei voi vaikuttaa hyökkäyksen aikana, mikä tuo peliin strategisen elementin puolustuksien sijoittelun muodossa. Kuva 8 esittää 32. tason pelaajan tukikohtaa. Muiden pelaajien planeettoja vastaan hyökätään erilaisilla roboteilla, joita pelaaja voi rakentaa siihen tarkoitetulla rakennuksella. Käytettävien robottien määrä ja tyyppien määrä kasvavat kokemustasojen myötä. Jokainen robottityyppi on mekaniikaltaan erilainen, mikä auttaa pelaajia taistelemaan erilaisia puolustuksia vastaan ja tuo peliin vaihtelevuutta.

Kuva 8. Nykyinen Last Planets näyttää hyvin erilaiselta edeltäjäänsä verrattuna.

Peli on suunniteltu pitämään sosiaalinen kanssakäynti ja pelaaminen keskipisteenä. Se kannustaa pelaajia pelaamaan keskenään ja pyrkii antamaan siihen tarvittavat työkalut. Pelaajat voivat muodostaa yhteisöjä, joita kutsutaan liittoumiksi. Liittoumat voivat rakentaa niille ominaisia rakennuksia, jotka hyödyntävät kaikkia siinä pelaavia pelaajia, sekä taistella muita liittoumia vastaan ja ottaa osaa ainoastaan liittoumille tarkoitetuissa tapahtumissa. Liittoutumassa pelaavat pelaajat voivat keskustella muiden liittouman jäsenten kanssa omalla keskustelukanavalla, joka on avoin ainoastaan liittouman jäsenille. Pelissä on myös keskustelukanava yleiselle keskustelulle sekä yksityisille keskusteluille.

Kokonaisuudessaan peli koostuu kolmesta eri näkymästä, joita ovat planeettanäkymä, galaksikartta ja taistelunäkymä. Planeettanäkymässä näkyy pelaajan oma tukikohta, jonka keskellä sijaitsee planeetta ja sen ympärillä pelaajan rakentamat rakennukset, kuten kuvassa 8 näkyy. Tässä näkymässä pelaaja pystyy rakentamaan, päivittämään ja liikuttamaan omia rakennuksia. Näkymän käyttöliittymästä pelaaja pystyy avaamaan

tapahtumalokin, asetukset-ikkunan, keskustelukanavaikkunan ja päivittäistehtäväikkunan (daily quest). Käyttöliittymän oikeassa nurkassa sijaitsee painike, jolla pelaaja pääsee vaihtamaan näkymän galaksikarttaan. Käyttöliittymän vasemmassa nurkassa sijaitsee painike, jota painamalla rakennettavien rakennusten valikko tulee näkyviin.

Kaikkien pelaajien planeetat sijaitsevat samassa maailmassa, jota pelin sisällä kutsutaan galaksiksi. Tässä galaksinäkymässä sijaitsevat kaikkien pelaajien planeetat ja heidän muodostamansa liittoumat. Planeettojen ja liittoumien paikat ovat staattisia. Galaksikarttaan on suunniteltu ominaisuus, jossa epäaktiiviset pelaajat siirtyvät hitaasti galaksin reunoille ja aktiiviset galaksin keskelle, mutta sitä ei ole vielä otettu käyttöön priorisoinnin takia. Pelaaja voi hyökätä mitä tahansa galaksikartasta löytyvää pelaajaa vastaan kokemustasoeroista riippumatta. Jos ero on tarpeeksi suuri tilanteissa, joissa korkeamman tason pelaaja hyökkää alhaisen tason pelaajaa vastaan, voitosta saadut resurssit ovat määrältään huomattavasti pienempiä. Hyökkäyskohteiden lisäksi galaksikarttaa käytetään liittoutumien löytämiseen ja niihin liittymiseen. Liittoutumaan voi liittyä erillisestä liittoumaikkunasta, jonka voi avata painamalla liittouman keskellä olevaa aurinkoa.

Ihmispelaajien lisäksi galaksikartassa sijaitsee pelin omia vihollisplaneettoja, joita ei omista oikea pelaaja. Näitä planeettoja kutsutaan BOTS-planeetoiksi, joiden puolustuksen rakenne on tehty etukäteen. Jokaista pelaajan kokemustasoa kohden on olemassa useita eri versioita BOTS-planeetasta. Planeetan valloittamisesta pelaaja palkitaan resursseilla, joita pelissä käytetään rakennusten rakentamiseen ja päivittämiseen. Vihollisplaneettojen vaikeustaso ja monimutkaisuus nousevat, mitä korkeammalla kokemustasolla pelaaja on. Vaikeustasoissa on silti varianssia, vaikka niiden taso olisi sama. Tämä heijastuu voitosta saataviin resursseihin.

Last Planets on rajoitettu yli 7-vuotiaille, mutta se on suunniteltu vetoamaan mahdollisimman suureen yleisöön. Tämä on otettu huomioon etenkin graafisen tyylin suunnittelussa. Projektin alussa taidetyyli määriteltiin yksinkertaiseksi kolmiulotteiseksi futuristiseksi tyyliksi, joka visuaalisesti muistuttaisi lasten lelulaatikkoa. Liian tummia värejä ja teräviä tai monimutkaisia muotoja ei saa käyttää, jotta peli ei vaikuttaisi liian vaikeasti lähestyttävältä. Projektin edetessä kokonaistyyli ei ole muuttunut, mutta pieniä muutoksia on tehty.

Peli sisältää runsaasti graafisia elementtejä ja erikoistehosteita. Pelissä on yhteensä 22 erilaista rakennusta, joita pelaaja voi rakentaa oman planeettansa ympärille. Jokaisella näistä rakennuksista on keskimäärin 15 eri tasoa, joihin ne voi päivittää. Rakennuksen visuaalinen ulkonäkö muuttuu lähes joka tasolla kehityksen alkuvaiheessa ja hidastuu loppua kohti. Tämä tarkoittaa sitä, että yhteensä erillisiä spritejä pelkille rakennuksille on yli 150. Määrä on todellisuudessa korkeampi, sillä esimerkiksi puolustukseen käytettävät tykkirakennukset (turret) tarvitsevat kahdeksan kuvaa joka 45 asteen käännökselle. Pelissä on myös muutamia 3D-malleja, jotka toimivat reaaliajassa. Näitä ovat muun muassa pelaajan suojelema planeetta ja vihollistukikohtien BOTS-planeetta.

5 Last Planets -pelin elementtien tuotantolinja

5.1 Unity pelimoottorina

Unity 3D on pelimoottori, jonka pääominaisuuksia ovat helppokäyttöisyys, aloittelijaystävällisyys ja monialustajulkaiseminen. Ensimmäinen Unityn versio julkaistiin vuoden 2005 heinäkuussa ja siitä on ajan myötä muodostunut yksi suosituimmista kehitysympäristöitä mobiili- ja pc-pelien kehitykseen. Unity ja kaikki sen ominaisuudet ovat käytössä sen ilmaisversiossa. Tapauksissa, joissa Unityn kautta julkaistu peli tuottaa yli 100 000 dollaria, yrityksen tai yksilön täytyy ostaa maksullinen lisenssi. [30.]

Kehitysympäristönä Unity on helppokäyttöinen ja nopea, sillä se sisältää useita valmiita osia ja toimintoja, joita ei tarvitse kehittää erikseen. Esimerkiksi pelielementin tuominen pelimoottorin editoriin ja sen saattaminen käyttövalmiiksi ei vaadi suurta työmäärää. Elementin raahaaminen editorin ikkunaan kopioi sen Unityn projektikansioon, minkä jälkeen se on vapaasti käytettävissä kehitettävässä pelissä. Elementin sisääntuonnin asetuksia voi muuttaa koska tahansa ilman lähdetiedoston muuttamista, mikä on erittäin hyödyllistä etenkin silloin, kun testataan eri pakkausalgoritmien vaikutusta lopputuloksen laatuun.

Editorin käyttöliittymä on selkeä ja sitä voi muokata tarpeen vaatiessa. Rakenteeltaan se muistuttaa monien pelien kenttäeditoreja. Ikkunan keskellä sijaitsee editorin näkymäikkuna, josta näkee, mitä skenessä on näkyvissä. Ikkunan yläpalkissa sijaitsevat valikot, kuten edit, file, window ja niin edelleen, jotka ovat tuttuja käyttäjälle useiden Windows-pohjaisten ohjelmien kautta. Ikkunan alareunassa on projektin tiedostoselain, josta kaikki projektissa olevat tiedostot voidaan löytää manuaalisesti selaten tai etsintätyökalun avulla. Oikeassa reunassa sijaitsee osio, josta voi nähdä valitun objektin tai elementtien ominaisuudet, kuten sijainnin 3D-avaruudessa, siihen liitetyt skriptit ja niin edelleen.

Editorin samankaltaisuus muiden editoreiden kanssa ja johdonmukainen työnkulku on mahdollistanut muidenkin kuin ohjelmoijien osallistumisen elementtien käyttöönottoon Last Planet -pelin kehityksessä, mikä on helpottanut koko työryhmän työskentelyä huomattavasti. Kun pelielementin tuo peliin itse elementin tekijä, virheiden määrä on

huomattavasti pienempi, sillä työtä ei tarvitse delegoida eteenpäin samalla altistaen sitä virheelliselle kommunikaatiolle tai ohjeille.

Alkuperäinen syy Unity 3D:n valitsemiseen Last Planetsin pelimoottoriksi oli sen tuki useille julkaisualustoille. Tällä hetkellä suurimmat mobiilimarkkinat ovat Android- ja IOS-pohjaisille laitteille, joten valinta tehdä peli molemmille markkinoille sopivaksi oli kannattavaa. Kääntöjärjestelmän tai wrapperin rakentaminen pelin julkaisemiseksi on aikaa vievää ja ongelmallista, joten Unityyn vaihtaminen aikaisemmin käytetystä LibGDX-java-ohjelmistokehyksestä oli järkevää. Unityn mahdollistama nopea kehitystahti ja hyödyllisien lisäosien saatavuus ovat toimineet hyvin projektin tarpeisiin.

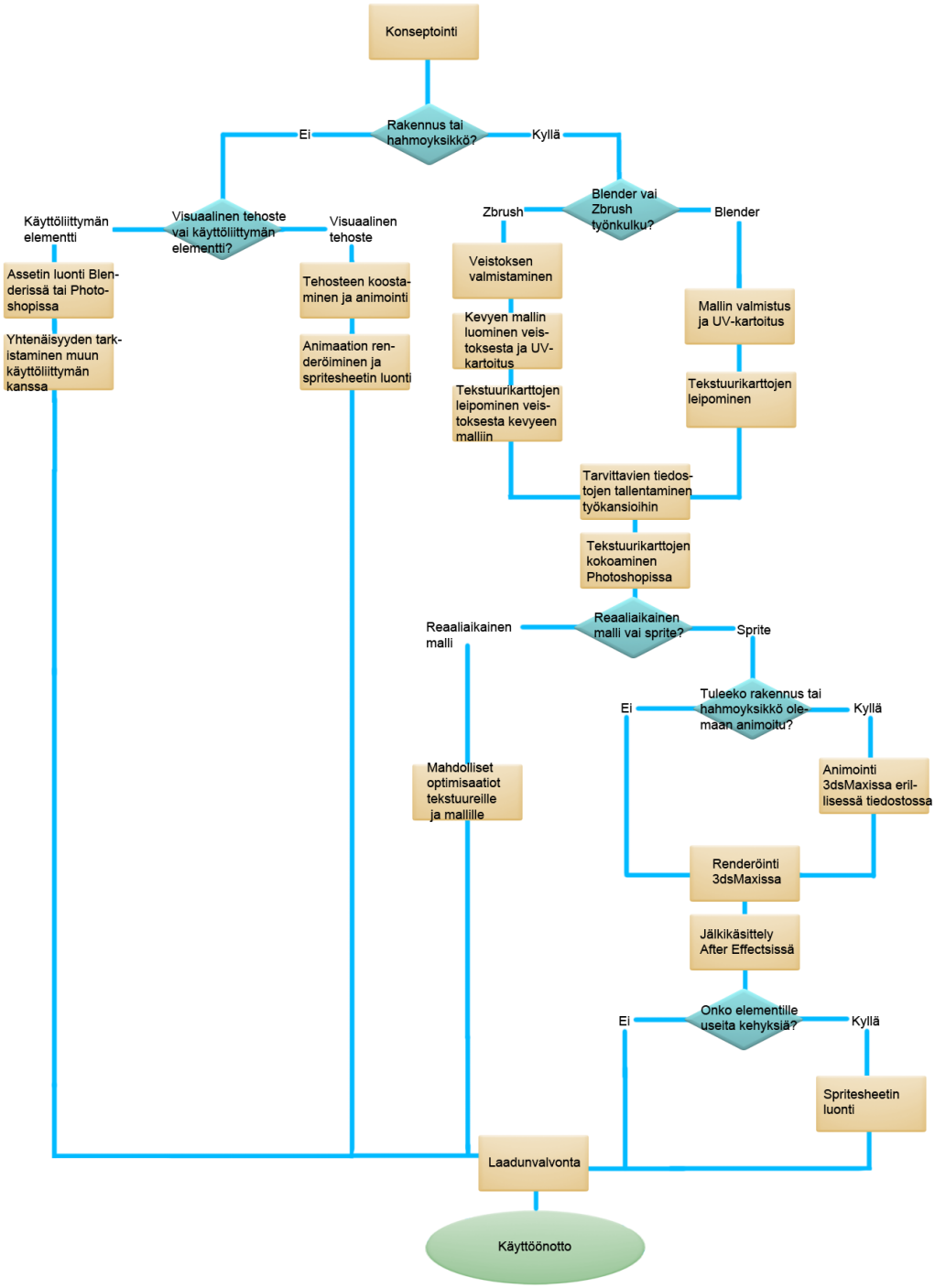
Vaikka Unity on helppokäyttöinen ja nopea kehitysympäristö, sillä on myös heikot puolensa. Samalla kun monet valmiit komponentit ja toiminnot ovat hyödyllisiä ja nopeita käyttää, niiden muokkaaminen projektin tarpeisiin ei ole välttämättä mahdollista. Tästä hyvänä esimerkkinä on automaattinen sprite packer -pakkain, joka tallentaa pelissä käytetyt spritet atlakseen automaattisesti. Sen toiminta ei vastannut projektin vaatimuksia, sillä se lisäsi yhden läpinäkyvän pikselin kaikkien spritejen reunuksien ympärille. Pakkaajan käyttämään logiikkaan ei ollut pääsyä, mikä käytännössä esti ongelman ratkaisemisen ja johti kompromissiin. Tämän lisäksi Unity on suhteellisen raskas pelimoottori, ja sen optimointi ei ole mahdollista yhtä pitkälle kuin itse tehdyssä moottorissa. Ongelmien määrä on kuitenkin huomattavasti Unityn tuomien etujen määrää pienempi, ja se on käytännössä paras vaihtoehto Last Planetsin kokoiselle projektille.

5.2 Tuotantolinjan rakenne

Last Planets -pelin graafisten pelielementtien tuotantolinja rakennettiin projektin tarpeiden ympärille. Sen ominaisuudet ja työvaiheet ovat muuttuneet projektin ketterän kehitysmallin mukana. Kaikki pelissä olevat rakennukset ja hahmot ovat kulkeneet läpi tuotantolinjan, jonka toimintaa voi seurata kuvasta 9. Tuotantolinjan työvaiheita ovat tällä hetkellä

- konseptointi
- mallinnus
- teksturointi

- animointi
- renderöinti
- jälkikäsittely
- käyttöönotto.

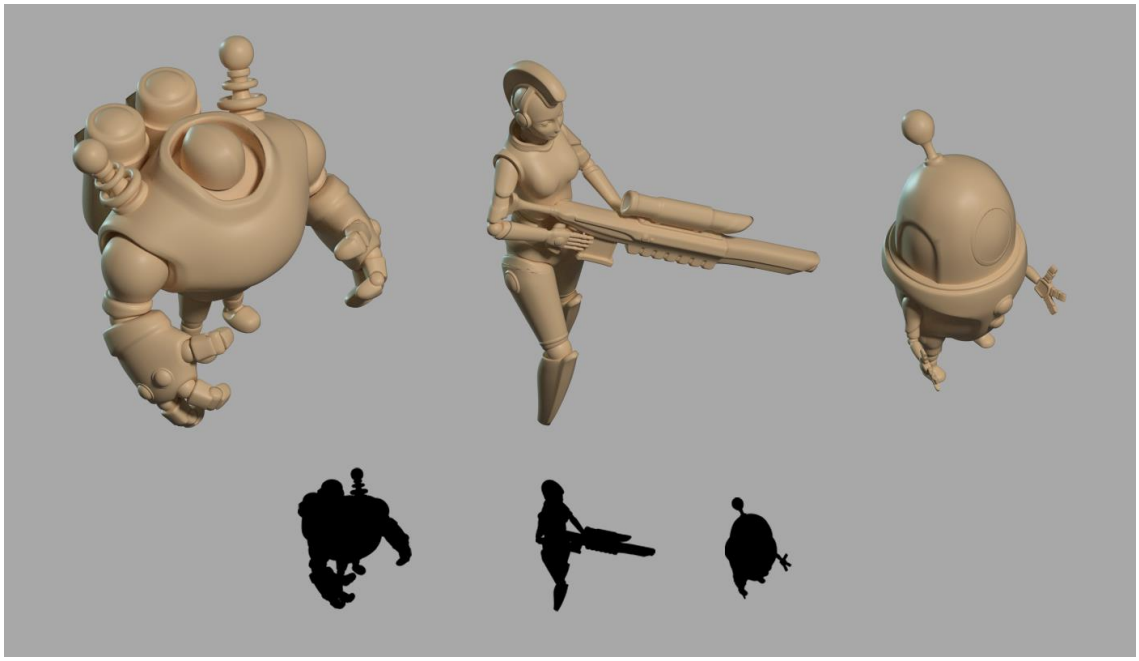


Kuva 9. Last Planets -pelin tuotantolinjan toiminnan prosessikaavio.

Konseptointi on tuotantolinjan ensimmäinen askel, jossa tuotettava elementti suunnitellaan. Alustavat ohjeet tarvittavalle elementille tulevat pelin suunnittelijalta kirjallisessa muodossa kirjoitettuna suunnitelmadokumenttiin tai suullisesti, jos suunnittelija ja pelielementin tekijä työskentelevät samassa tilassa. Yleisesti suunnittelijan antamat ohjeet ovat suuntaa antavia eivätkä määritä tarkkoja muotoja, ellei niillä ole merkittävää vaikutusta rakennuksen tai hahmon toimintaan pelissä. Konseptitaide on projektissa tarkoitettu ainoastaan sisäiseen käyttöön. Pääasiana on muotojen ja perusidean esille tuonti viimeistellyn lopputuloksen sijaan. elementistä riippuen konseptointi voi kestää noin kahdesta tunnista jopa kuuteen tuntia.

Rakennuksia ja muita monitasoisia pelin elementtejä suunnitellessa täytyy muistaa, että ne muuttuvat visuaalisesti eri tasoilla. Esimerkiksi pelin puolustuksen perusyksiköllä, Turretilla, on yhteensä noin seitsemän visuaalista muotoa, jotka konseptin tekijän tulee suunnitella. Tämä on haastavaa, sillä rakennus ei saa muuttua liikaa tasojen välillä ja sen täytyy mukailla graafista ohjeistoa. Hyväksi apukeinoksi on todettu työnkulku, jossa aluksi suunnitellaan päätasot, joita esimerkiksi 15-tasoisen rakennuksen suunnitelmassa on kolme. Kun päätasot ovat valmiit, niiden väliin suunnitellaan jäljelle jäävät tasot muodostaen parhaassa tapauksessa saumattoman etenemisen tasolta tasolle ilman liian isoja muutoksia rakennuksen ulkonäköön kerralla. Visuaalinen muutos tasojen välissä ei saa kuitenkaan olla liian pieni, sillä rakennuksen päivittämisen täytyy tuntua palkitsevalta ja merkitykselliseltä.

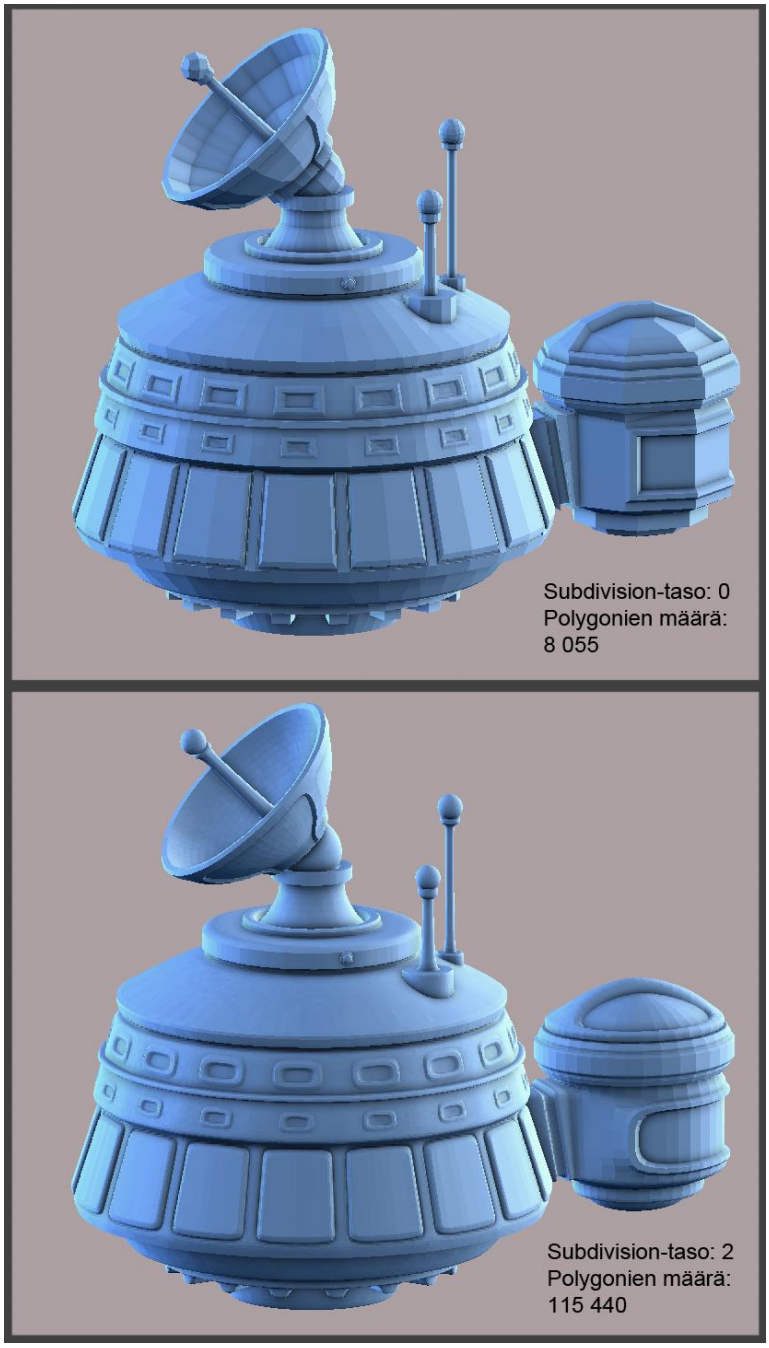
Hahmoyksiköillä ei ole visuaalisesti muuttuvia tasoja, mutta ne ovat haastavia muilla tavoilla. Pelissä käytettävien hahmojen täytyy täyttää neljä kriteeriä, jotta niitä voidaan käyttää pelissä. Hahmon tulee olla helposti lähestyttävä ja samaistuttava. Tämä yleensä tehdään antamalla hahmolle kasvot tai kasvojen kaltaisia elementtejä, joiden kautta se pystyy ilmaisemaan erilaisia tunnetiloja ja eleitä. Hahmon ulkonäön tulee välittää pelaajalle mielikuva, mitä se pelissä tekee ja mihin sitä käytetään. Erilaisten elementtien, kuten aseiden tai varusteiden, avulla voidaan välittää tämä tieto pelaajalle. Tästä esimerkkinä on pelin hahmo Mega, joka on lähitaisteluun keskittynyt yksikkö. Sen toimintaa pelissä edustaa sille annettu suuri vasara. Eri hahmojen tulee erota toisistaan tarpeeksi hyvin, sillä hahmojen koko näytöllä on etenkin hyökkäystilanteessa pieni. Tässä suurena tekijänä ovat hahmojen siluetit, joiden avulla niiden tunnistaminen helpottuu huomattavasti, mikä on huomattavissa kuvassa 10. Silueteista muodostuu usein automaattisesti tarpeeksi erilaisia, mutta ajoittain niitä joudutaan korjaamaan eroavaisuuden lisäämiseksi.



Kuva 10. Hahmoyksikköjen eroavaisuudet luovat niille uniikin siluetin, joka auttaa niiden hahmottamista pienessä koossa.

Kun konseptointikuva on valmis ja sisältää kaiken tarvittavan tiedon pelielementistä, sen 3D-mallinnus voidaan aloittaa. Projektin kehityksessä on käytetty kahta eri mallinnustapaa. Ensimmäinen mallinnustapa suoritetaan ainoastaan Blender-ohjelmistolla subdivision-mallinnusta käyttäen. Toinen käytetty tapa on tehdä ZBrush-ohjelmistossa 3D-veistos, josta luodaan alhaisemman resoluution versio. Vaikka kahden erillisen tavan käyttö on periaatteessa tuotantolinjaa hidastava tekijä, on todettu, että sen vaikutus elementtien tuottamiseen on tällä hetkellä vähäinen. On mahdollista, että tuotantolinjaa yhtenäistetään myöhemmin kehityksessä. Mallinnusvaihe kestää keskimäärin noin 6–8 tuntia, mutta elementtikohtaisia viivästyksiä saattaa tapahtua.

Subdivision-mallinnus Blenderillä on nopeaa ja epädestruktiivista. Mallin resoluutiota kasvatetaan subdivision-muokkaajalla, jonka kerrointa voi nostaa ja laskea tarpeen mukaan. Subdivision-mallinnus tuottaa menetelmänä pelin graafisen ohjeiston mukaisia pyöreitä muotoja lähes automaattisesti ja mahdollistaa yksityiskohtien tarkkuuden skaalauksen resoluution avulla, kuten kuvasta 11 voidaan havaita. Toimiakseen hyvin subdivision-mallinnus vaatii hyvän suunnitelman elementistä konspetina, sillä mallinnustyylin laatu on sidonnainen mallin rakenteen laatuun. Mallintajan tulee myös miettiä tarkkaan, miten erilaiset muodot kannattaa toteuttaa, jotta mallin rakenne tukee siihen tulevia yksityiskohtia. Koska mallit eivät tule peliin 3D-malleina vaan kuvina, subdivision-mallinnuksen tuottamat korkean resoluution mallit eivät haittaa kehitystä.



Kuva 11. Subdivision-mallinnuksella valmistettu etuvartiorakennus.

Zbrush-ohjelmaa käyttävällä tavalla mallintamisprosessi pitenee yhdellä askeleella, mutta tarjoaa samalla enemmän vapautta. Blenderiä käytetään karkean 3D-mallin luomiseen, joka muodostuu erillistä osista. Nämä osat tallennetaan obj-muodossa työskentelykansioon, josta ne ladataan Zbrushiin. Zbrush on 3D-mallinnus- ja veistosohjelma, joka mahdollistaa erittäin korkearesoluutioisten mallien veistämisen erilaisilla työkaluilla. Veistämistyönkulku mahdollistaa 3D-mallien luonnin ilman, että tekijän tarvit-

see välittää mallin rakenteesta ja siihen vaikuttavista muokkaajista. Kun veistos on saatu valmiiksi ohjelmassa, se tallennetaan työkansioon obj-muodossa.

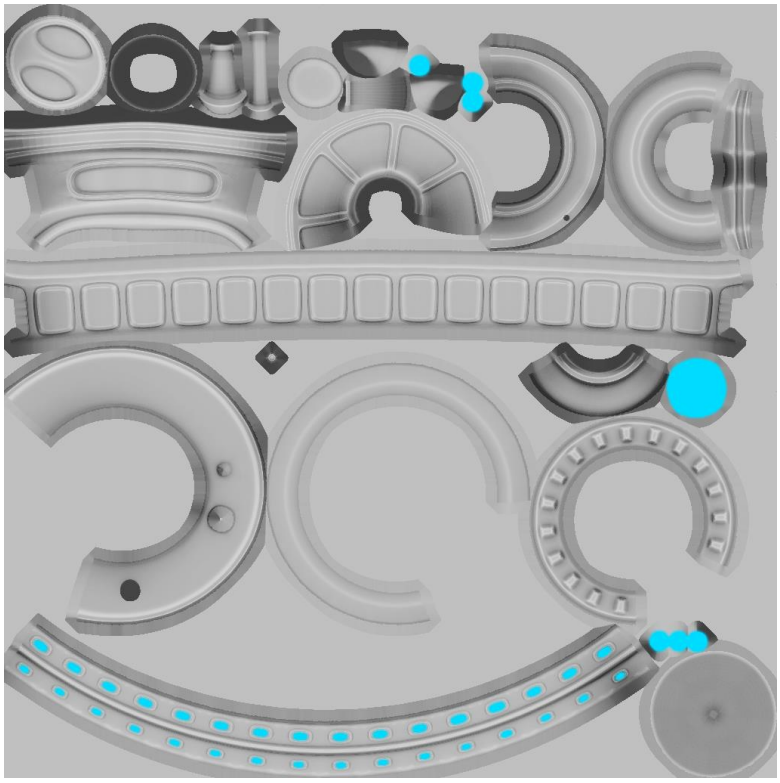
Veistettyä 3D-mallia ei kannata käyttää renderöintiin sellaisenaan sen suuren resoluution takia. Tästä syystä veistoksesta täytyy tehdä pienemmän polygonimäärän versio, jonka tekstuureihin veistoksen pinnan tiedot tallennetaan normaalikarttana. Korkearesoluutioisen veistoksen informaatio tallennetaan alemman resoluution malliin Xnormal-ilmaisohjelmalla. Ohjelman laskemat tekstuurit tallennetaan erilliseen tekstuurikansioon. Pelielementin mallinnus veistämällä vaatii enemmän työvaiheita lopputuloksen aikaansaamiseksi, mutta se tarjoaa enemmän vapautta mallinnukseen. Etenkin monimutkaisempia malleja on helpompi mallintaa tällä tavalla, sillä mallin rakennetta ei tarvitse miettiä erikseen.

Jos elementille on suunniteltu animaatioita, niiden vaatimukset täytyy ottaa huomioon mallinnusvaiheessa. Tämä on erityisen tärkeää esimerkiksi hahmoissa, joiden raajojen täytyy pystyä liikkumaan tarpeeksi halutun animaation onnistumiseksi. Kohdat, joista malli taittuu tai venyy, tarvitsevat enemmän geometriaa toimiakseen oikein. Tällä hetkellä suuri osa animaatioista toteutetaan renderöimällä koko animaatio erillisinä kuvina, joista koostetaan yksi iso kuva, jota pelimoottori käyttää animaation toistamiseen. Vaikka esilasketut animaatiot antavat paljon vapautta animointiin, sen luomat suuret kuvat vievät paljon välimuistia kohdelaitteella. Tästä syystä animaatiot tullaan tulevaisuudessa tekemään vasta pelimoottorissa erillisistä paloista resurssien säästämiseksi. Tämän lisäksi ne vievät paljon aikaa. Animaatioiden tekeminen voi kestää jopa päiviä riippuen animaatioiden määrästä ja laadusta.

Riippumatta siitä, kummalla tavalla malli on mallinnettu, sillä tulee olla UV-kartta määritettynä tekstuureita varten sekä geometriasta lasketut tekstuurit. Teksturoinnin kannalta olennaista on, että mallille on olemassa siitä laskettu ambient occlusion -kartta, joka kuvaa taustavalon vaikutusta malliin.

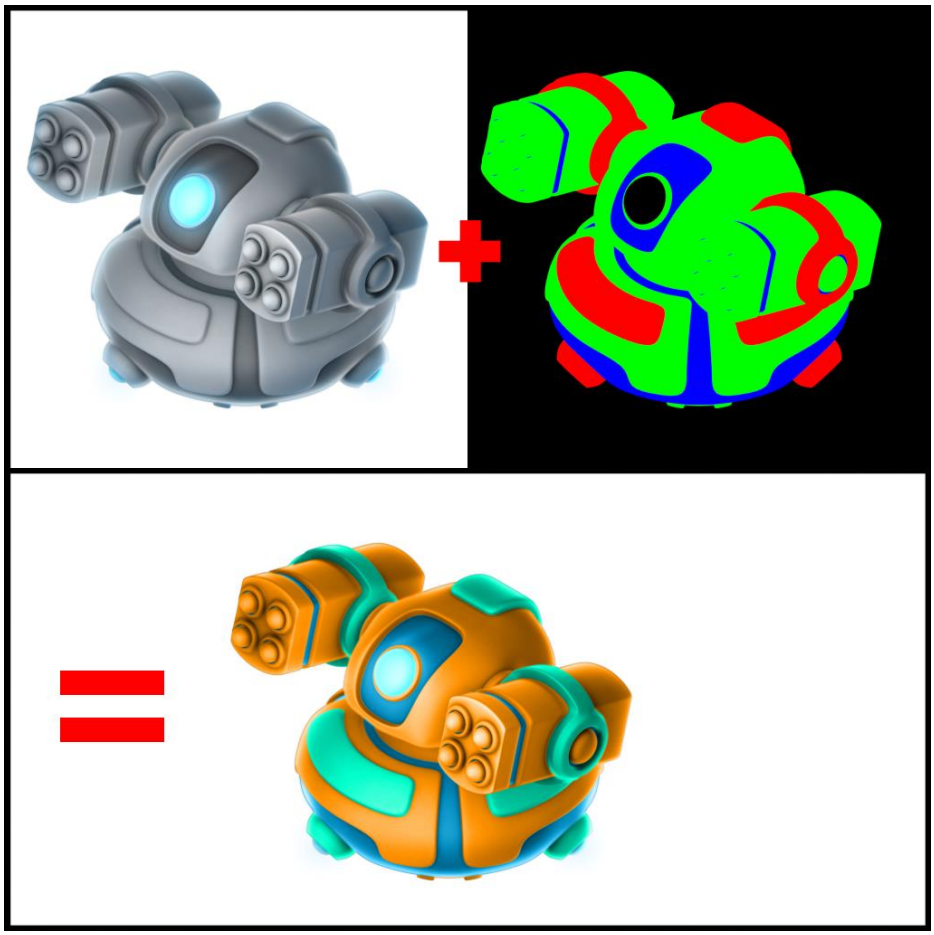
Mallit teksturoidaan enimmäkseen Adoben Photoshopissa. Se on tuotantolinjan nopein työvaihe. Sen suorittaminen kestää arviolta tunnin. Tekstuureille on tehty yhtenäinen päätiedosto, joka kopioidaan elementin omaan työkansioon. Päätiedosto on Photoshop-dokumentti, joka sisältää tarvitun tasorakenteen (layer structure) sekoitustiloineen. Elementin tekijän tarvitsee ainoastaan korvata tiedostossa olevat tasot mallista lasketuilla tekstuureilla. Tasot sisältävät yleisesti ottaen pohjavärin, ambient occlusion

-kartan, mallin reunojen vaalennuksen ja erilaisten valojen värin. Lopputuloksena saatu tekstuuri on pääosin harmaasävytteinen, mikä on nähtävissä kuvassa 12.



Kuva 12. Etuvartiorakennuksen diffuusiotekstuuri.

Graafiset pelielementit värjätään pelimoottorissa kolmannen osapuolen lisäosan avulla. Tämä lisäosa käyttää RGB-kuvatekstuuria värjäten kunkin kanavan halutulla värillä, mikä mahdollistaa lukemattomien eri väriyhdistelmien käytön rakennuksille. Esimerkki tästä on nähtävissä kuvassa 13. Lisäosa pidentää kehysaikaa vanhemmilla kohdelaitteilla, minkä takia sen tilalle joudutaan etsimään muita ratkaisuja. Toistaiseksi lisäosan käyttämä tekstuuri on osa tuotantolinjaa ja se koostetaan samassa päätiedostossa.



Kuva 13. Pitkän kantaman puolustusrakennuksen sprite, sen RGB-tekstuuri ja niiden värjäyksen lopputulos pelimoottorissa color-sekoitustilalla.

Tällä hetkellä tuotantolinjassa käytetään ainoastaan diffuusiotekstuuria ja väritys-lisäosan tekstuuria. Aikaisemmin mukana olivat muun muassa tekstuuri spekulaariheijastuksen värille, mutta niiden vaikutus lopputulokseen koettiin marginaaliseksi, minkä takia ne poistettiin kokonaan työvaiheesta. Zbrushissa tehdyt mallit vaativat toimiakseen normaalikartan, mutta se ei vaadi erillistä teksturointia.

Reaaliajassa toimivat mallit teksturoidaan päätiedostosta riippumattomina, vaikka samankaltaista tasorakennetta voidaan käyttää. Teksturointiin voidaan käyttää Photoshopin lisäksi Blenderin 3D-maalausta, jos elementin UV-kartoitus estää helpon havainnollistamisen maalattavasta alueesta. Jos peliin suunnitellaan suurempia määriä reaaliajassa toimivia 3D-malleja, niille täytyy järjestää oma osio tuotantolinjaan.

3D-mallin ja tekstuurien valmistuttua elementti ladataan Autodesk 3DsMax -ohjelmaan, jolla se renderöidään kuvaksi. Samalla tavalla kuin teksturoinnissa, päätiedosto on en-

nalta rakennettu, mikä helpottaa työvaihetta huomattavasti. Se sisältää kaikki elementin renderöimiseen tarvittavat komponentit, kuten valot, materiaalit ja editorin asetukset. Kun tiedosto on kopioitu työkansioon, elementin malli tuodaan ohjelmaan sisään. Ensimmäiseksi 3D-objektia joudutaan suurentamaan moninkertaiseksi, sillä 3DsMaxin ja Blenderin yksikköjärjestelmät eivät ole samat. Suurennoksen jälkeen elementille annetaan ennalta rakennettu materiaali, johon eri tekstuurit liitetään. Renderöimisvaihe kestää noin tunnin, mutta animoitujen mallien valmistelu ja renderöinti kestää huomattavasti pidempään.

Pelielementistä ei renderöidä ainoastaan yhtä kuvaa, vaan sen komponentteja, jotka myöhemmin sekoitetaan jälkikäsittelyohjelmassa säädettävän lopputuloksen saamiseksi. Tästä syystä päätiedostossa käytetään render-tasoja (render state), jotka mahdollistavat useiden materiaalien käytön samalle objektille sekä erilaisten asetusten käyttöä jokaista tasoa kohti. Kun materiaalit ja asetukset ovat määriteltynä, kaikki render-tasot voidaan lähettää Microsoftin Azure -palvelussa toimivalle palvelimelle, joka kykenee renderöimään kaikki tasot nopeasti. Palvelin tallentaa tiedostot elementin työkansioon automaattisesti niiden valmistuttua.

Animaatiota tarvitseva pelielementti animoidaan 3DsMaxissa erillisessä tiedostossa, joka linkitetään päätiedostoon, ja se tallennetaan eri versiona, mikä on määritelty tiedostonimessä. Animaatiot ovat yleensä 8, 16, 24, 32 tai 64 kuvaa pitkiä riippuen kuvien lopullisesta koosta pelissä. Koska kuva-animaatiot vievät paljon muistia, niiden määrä pyritään pitämään mahdollisimman pienenä.

Kun palvelin on saanut kaikki kuvat tallennettua työkansioon, tuotantolinjan viimeisenä vaiheena toimiva jälkikäsittely voidaan aloittaa. Jälkikäsittelyyn käytetään Adoben After Effects -videonkäsittelyohjelmaa, joka toimii hyvin myös kuvien kompositointiin ja säätämiseen. Jälkikäsittely aloitetaan kopioimalla ennalta rakennettu päätiedosto elementin työkansioon. Työnkulku toimii samalla tavalla kuten teksturoidessakin. Tiedostossa olevat kuvat korvataan yksi kerrallaan 3DsMaxin renderöimillä kuvilla. Uudet kuvat päivittyvät automaattisesti jokaiseen kohtaan dokumentissa, missä niitä on käytetty. After Effects -tiedosto koostuu useista precomposition-kansioista, jotka ovat verrattavissa Photoshopin kansioihin. Niille voi asettaa erilaisia sekoitustiloja ja tehosteita, ja niitä voi käyttää maskina muille precomposition-kansioille.

Pääasiallisesti After Effectsissä tehtävä kompositio littää kaikki renderöidyt kuvaelementit yhdeksi kokonaiseksi kuvaksi ennalta määrätyillä arvoilla, joita voi säätää tilannekohtaisesti. Tämän lisäksi erilaisten tehosteiden lisääminen kuvaan tehdään siinä. Monia visuaalisesti näyttäviä tehosteita on vaikea toteuttaa hallitusti 3D-ohjelmistossa, minkä takia ne kannattaa tehdä vasta jälkikäsittelyssä. Tästä hyvänä esimerkkinä toimivat pelissä olevien rakennusten valojen hehkut ja piirrosmaiset heijastukset. Kun kompositio on valmis, lopputulos renderöidään ulos ohjelmasta halutulla resoluutiolla, joka esimerkiksi rakennuksille on 256 x 256. Valmis kuva tallennetaan työkansion ulkopuolelle, ja se nimetään projektin sisäisen nimeämiskäytännön mukaan. Yksittäiset elementit valmistuvat nopeasti jälkikäsittelystä. Arvioitu keskimääräinen työvaiheen kesto on enintään kaksi tuntia, ellei kyseessä ole elementti, jolla on monta kehystä.

Valmis sprite käyttöönotetaan Unityssä kahdessa osassa. Ensin elementti tuodaan projektiin, mikä tehdään yksinkertaisesti raahaamalla tiedosto Unityn ikkunaan, jossa haluttu kansio projektista on auki. Tämän jälkeen kuvalle määritetään halutut import-asetukset, jotka vaikuttavat kuvan laatuun, tyyppiin, kompressioon ja kokoon pelissä pixels per unit -arvon kautta. Spriten koko on ainoa muuttuva tekijä import-asetuksissa, sillä siihen vaikuttaa sille varattu tila pelissä, sekä sen ulkonäkö.

Kun elementille on säädetty oikeat import-asetukset, se jätetään projektiin sellaisenaan odottamaan käyttöönottoa, jonka tekee ohjelmointiryhmä. Jos valmistettu elementti on uusi sprite valmiiksi käyttöönotetulle rakennukselle, se voidaan tehdä muokkaamalla rakennuksen prefabia, joka on esivalmistettu kokonaisuus Unityssä. Prefabin sisältä löytyy objekti, joka on nimetty rakennuksen nimellä ja siihen on lisätty Sprite Renderer -komponentti. Sen käyttämän spriten vaihtaminen korvaa rakennuksen käyttämän spriten, minkä jälkeen koko prefab tulee tallentaa painamalla Apply-painiketta.

5.3 Muutokset ja haasteet

Vaikka tuotantolinjan määrittelisi mahdollisimman tarkasti alusta alkaen, on todennäköistä, että se muuttuu projektin edetessä. Ketterää kehitysmallia käytettäessä odottamattomia muutoksia joudutaan tekemään, mikä usein saattaa johtaa ongelmiin. Kompromissien teko on osa pelinkehitystä, ja siihen on sopeuduttava. Parhaassa mahdollisessa tilanteessa tuotantolinjan rakennetta joudutaan muuttamaan vain vähän kerral-

laan ja käytettyjä ohjelmia ja käytäntöjä ei tarvitsisi muuttaa kertaakaan. Näin ei ole käynyt Last Planetsin kanssa.

Last Planets on ollut projektina hyvin haastava jokaiselta osa-alueeltaan. Tämä yhdistettynä työryhmän kokemattomuuteen projektin alussa on taannut, että osa asioista on tehty väärin tai vaikeimman mahdollisen tavan kautta. Pelin tuotantolinja on muuttunut merkittävästi yli neljään kertaan, ja jokaisella kerralla muutosten määrä on ollut suuri. Kun projekti aloitettiin, tuotantolinjan muodostamista ei otettu lainkaan huomioon, pelielementtejä tehtiin peliin hyvin vapaasti ilman ennalta määrättyjä arvoja tai toimintamalleja. Tässä vaiheessa projektissa käytettiin vain Blenderiä ja jälkikäsittelyä ei tehty elementeille ollenkaan.

Myöhemmässä projektin vaiheessa renderöintiohjelmaksi valittiin Autodesk Maya, joka vastasi projektin tarpeita hyvin. Sen kanssa ilmeni kuitenkin ongelmia, kun verkkorenderöintiä yritettiin saada toimimaan tuotantolinjassa. Tämä pakotti tuotantolinjan siirtymään Mayasta 3DsMax-ohjelmistoon. Vaikka molemmat ohjelmistot ovat saman yrityksen hallinnoimia, ne eivät kuitenkaan tue toistensa tiedostoformaatteja. Kaikki elementit jouduttiin kääntämään 3DMaxin formaattiin käsin. Ohjelman vaihdosta oli kuitenkin osittain hyötyä, sille se auttoi taidetyöryhmää iteroimaan assetteja eteenpäin ja hyödyntämään aikaisempaa osaamista ohjelmasta.

Viimeisin suuri vastoinkäyminen tapahtui juuri ennen esijulkaisua. Pelissä käytetty kolmannen osapuolen lisäosa osoittautui pääsyyksi pelin alhaiseen kehysnopeuteen, mikä huomattiin optimoinnin yhteydessä. Lisäosaa käytettiin rakennusten värjäämiseen pelissä, mikä oli olennainen osa pelin ulkonäköä. Lisäosan skriptit ja shader-ohjelmat jouduttiin poistamaan pelin rakennuksista, mikä johti kaikkien rakennusten värjäämiseen manuaalisesti Photoshopissa. Kaikista lisäosaa varten valmistetuista elementeistä, kuten RGB-tekstuureista, tuli hyödyttömiä. Rakennusten värjäämiseen ei ole toistaiseksi löytynyt hyvää ratkaisua, mutta uusia vaihtoehtoja etsitään aktiivisesti.

Lisäosan aiheuttamat ongelmat ovat johtaneet tuotantolinjan uuden version kehittämiseen. Käytettävät ohjelmat pysyvät ennallaan, mutta esimerkiksi renderöintiin käytettyä materiaalikirjastoa pyritään uudistamaan parempien lopputulosten saamiseksi. Samalla rakennusten suunnitteluun liittyviä ohjeita pyritään parantamaan ja tarkentamaan. Tuotantolinja kokonaisuudessaan on kehittynyt merkittävästi alkutilanteesta ja alkaa saada

lopullista muotoaan. On kuitenkin oletettavaa, että pieniä ja suuria muutoksia tapahtuu työryhmän ponnisteluista huolimatta.

6 Yhteenveto

Nykymaailmassa tuotantolinjoja käytetään kaikkialla, ja niistä on muodostunut olennainen osa ihmisten maailmaa. Teknologian ja toimintamallien kehittyessä tuotantolinjojen tehokkuus on kasvanut globaalisti alasta riippumatta. Järjestäytyneen ja tarkkaan harkitun tuotantolinjan merkitys on korostunut etenkin ohjelmistokehityksessä, jonka osa peliteollisuus on.

Tuotantolinjojen rakenteet ovat merkittävässä roolissa projektin onnistumisen kannalta. Hyvin määritellyt ja tarkkaan harkitut tuotantolinjat helpottavat koko projektin kehittämistä ja iterointia, mikä ei koske ainoastaan elementtejä pelinkehityksessä. Tuotantolinjan tuotteena voi olla niin pelimekaniikan konsepti kuin ohjelman osakin. Tuotantolinjat toimivat koko projektin tukirankana, joka pitää kaikki niiden läpi kulkevat elementit, konseptit ja niin edelleen yhteensopivina toistensa kanssa. Tämä on mahdollista vain tilanteessa, jossa tuotantolinjojen määritteet ovat huolellisesti tehtyjä ja ne on pidetty ajan tasalla.

Peliteollisuudessa tuotantolinjoille on kehittynyt vakituinen paikka, ja jokaiselta isolta kaupalliselta projektilta odotetaan tuotantolinjojen tehokasta käyttöä. On mahdollista, että tulevaisuudessa tuotantolinjojen määrittelystä tulee entistä tärkeämpää, sillä tuotannossa käytetyn automatisoinnin määrä nousee tekniikan kehittyessä. Automatisoinnin ansiosta kehityksen nopeus kasvaa, mikä tehostaa tuotantolinjan tuottavuutta. Graafisten elementtien luonnissa automatisointi voi tarkoittaa esimerkiksi proseduraalisia tekstuurigeneraattoreita, joista hyvä esimerkki on Allegorithmicin kehittämä Substance Painter. Sen mahdollistama erittäin nopea teksturointi säästää paljon aikaa assettia tekevältä työntekijältä ja täten mahdollistaa työn nopeamman etenemisen.

Suunnitelma Last Planetsin tuotantolinjan muodostamisesta kokonaisuudessaan onnistui hyvin. Johdonmukaiset määritteet erilaisten graafisten pelielementtien valmistukseen ja yhteensopivat ohjelmistot toimivat hyvin ja mahdollistavat huomattavasti paremman ja yhtenäisemmän assettien kehityksen. Vaikka uusi tuotantolinja olikin huomattava parannus aikaisempaan verrattuna, kehityksen aikana esille nousseiden ongelmien määrä on luonut tarvetta uudelle päivitykselle. Pelimoottorin värjäyslisäosan poisto oli harmittava takaisku hyvin toimineelle visuaalliselle ilmeelle. Tämän lisäksi osaa pelin graafisesta ilmeestä haluttaisiin parannella, mikä osittain vaikuttaisi rakennusten tyyliin. Tuotantolinja ei ole siitä huolimatta epäonnistunut. Se toimii hyvänä poh-

jana jatkokehitykselle ja iteraatiolle, ja sen aikana opitut asiat eivät varmasti mene hukkaan.

Tulevaisuutta ei voi varmasti ennustaa, mutta on todennäköistä, että tuotantolinjat ja niissä käytetty teknologia kehittyvät jatkuvasti. Tästä syystä on tärkeää pysyä mukana kehityksessä ja seurata suuria muutoksia tekniikassa. Ajan tasalla pysyminen auttaa ongelmien ratkaisemisessa, sillä uusia ratkaisuja keksitään jatkuvasti.

Lähteet

1 Assembly line Methods. 2014. Verkkodokumentti. Inc. <http://www.inc.com/encyclopedia/assembly-line-methods.html>. Luettu 4.4.2016.

2 How to create a design style guide: 25 pro tips. 2013. Verkkodokumentti. Creative Bloq. < http://www.creativebloq.com/design/create-style-guides-1012963> Updated 5 July 2013. Luettu 4.3.2016.

3 Corday, Robert. 2014 The evolution of assembly lines: A brief history. Verkkodokumentti. Robohub. < http://robohub.org/the-evolution-of-assembly-lines-a-brief-history/>. Updated 24 April 2014. Luettu 3.3.2016.

4 The Hardest Part of Making Games. 2013. Verkkodokumentti. Game Dev Gone Rogue. <http://gamedevgonerogue.blogspot.fi/2013/07/the-hardest-part-of-making-games.html>. Updated 30 September 2013. Luettu 6.3.2016.

5 Van Dongen, Joost. 2007. The Game Asset Pipeline. Verkkodokumentti. EMMA Game Design & Development Utrecht School of the Arts. <http://www.proungame.com/Oogst3D/ARTICLES/TheGameAssetPipeline.pdf>. Luettu 6.3.2016.

6 Vivo Patricio, Gonzalez. 2015. The book of Shaders. Verkkodokumenti. <http://patriciogonzalezvivo.com/2015/thebookofshaders/01/>. Luettu 6.3.2016.

7 Clare, Adam. 2013. Mobile Optimization Tips for Unity. Verkkodokumentti. <http://www.realityisagame.com/archives/2116/mobile-optimization-tips-for-unity/>. Updated 14 August 2013. Luettu 8.3.2016.

8 Make better textures for games, 'Power of two' & proper image dimensions. 2016. Verkkodokumentti. Katsbits. <http://www.katsbits.com/tutorials/textures/make-better-textures-correct-size-and-power-of-two.php>. Luettu 10.3.2016.

9 Polygon count for smartphone applications. 2014. Verkkodokumentti. Polycount. <http://polycount.com/discussion/130371/polygon-count-for-smartphone-applications>. Luettu 12.3.2016.

10 Basic UV Mapping. 2015. Verkkodokumentti. ModoTutorials. <http://www.modonize.com/Communities/Members/1/Bob%20deWitt/UVMappingBasics.pdf>. Luettu 11.3.2016.

11 Uv mapping tips and tricks. 2004. Verkkodokumentti. Gamasutra. <http://www.gamasutra.com/view/feature/130484/uv_mapping_tips_and_tricks.php>. Luettu 12.3.2016.

12 Difference between mobile and pc development. 2012. Verkkodokumentti. Javatechig. < http://javatechig.com/gaming/mobile-game-dev-to-pc-game-dev>. Luettu 13.3.2016.

13 What's the Difference Between Raster and Vector? 2015. Verkkodokumentti. PRPrint. <https://www.psprint.com/resources/difference-between-raster-vector/>. Luettu 15.3.2016.

14 Raster vs Vector. 2014. Verkkodokumentti. Vector Conversions. <http://vector-conversions.com/vectorizing/raster_vs_vector.html>. Luettu 16.3.2016.

15 Kudler, Amanda. 2008. Timeline: Video Games. Verkkodokumentti. <http://www.infoplease.com/spot/gamestimeline1.html>. Luettu 18.3.2016.

16 Sprite Graphics. 1980. Verkkodokumentti. Commodore computers. <http://www.commodore.ca/manuals/c64_users_guide/c64-users_guide-06-sprite_graphics.pdf>. Luettu 19.3.2016.

17 The guide to implementing 2D platformers. 2012. Verkkodokumentti. Higher-Order Fun. <http://higherorderfun.com/blog/2012/05/20/the-guide-to-implementing-2D-platformers/>. Luettu 21.3.2016.

18 Sprite Renderer. 2016. Verkkodokumentti. Unity Technologies. <http://docs.unity3D.com/Manual/class-SpriteRenderer.html>. Luettu 24.3.2016.

19 Exporting 3D models as 2D sprites. 2013. Verkkodokumentti. Studica. <http://www.studica.com/blog/exporting-3D-models-as-2D-sprites>. Luettu 25.3.2016.

20 Using texturetool to Compress Textures. 2016. Verkkodokumentti. Apple. <https://developer.apple.com/library/ios/documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/TextureTool/TextureTool.html>. Luettu 27.3.2016.

21 Draw Call Batching. 2016. Verkkodokumentti. Unity Technologies. <http://docs.unity3D.com/Manual/DrawCallBatching.html>. Luettu 27.3.2016.

22 What is game design. 2013. Verkkodokumentti. International Student. <http://www.internationalstudent.com/study-game-design/what-is-game-design/>. Luettu 29.3.2016.

23 Practice: Do Designers Need To Be Programmers? 2011. Verkkodokumentti. Gamasutra. <http://www.gamasutra.com/view/news/127717/PRACTICE_Do_Designers_Need_To_Be_Programmers.php>. Luettu 30.3.2016.

24 Fallout 2 review. 2015. Verkkodokumentti. Old PC gaming. <http://www.oldpcgaming.net/fallout-2-review/>. Luettu 30.3.2016.

25 Cueto, Gerard. 2013. Six things you may not know about game development. Verkkodokumentti. <http://www.gamesradar.com/six-things-you-may-not-know-about-game-development-former-game-developer/>. Luettu 2.4.2016.

26 Schell, Jesse. 2008. The art of game design: Book of lenses. CRC Press.

27 Game Art Bible – Secret Sauce To Making Great Game Art. 2013. Verkkodokumentti. Pencillati Interactive. <http://www.slideshare.net/pencillati/game-art-bible-secret-sauce-to-making-great-game-art/18>. Luettu 2.4.2016.

28 Basic Sprite Tutorial. 2016. Verkkodokumentti. 2D will never die. <http://2Dwillneverdie.com/tutorial/a-basic-sprite-tutorial/>. Luettu 6.4.2016.

29 Game Tools Tune-Up: Optimize Your Pipeline Through Usability. 2009. Verkkodokumentti. Gamasutra. <http://www.gamasutra.com/view/feature/132407/game_tools_tuneup_optimize_your_.php>. Luettu 6.4.2016.

30 FAQ: Licensing and activation. 2016. Verkkodokumentti. Unity Technologies. < https://unity3d.com/unity/faq>. Luettu 24.4.2016.